Computer Networks Advanced Course

Lesson 3 – DNS and Scapy

Barak Gonen
Some slides from Jim Kurose

Lesson Goals

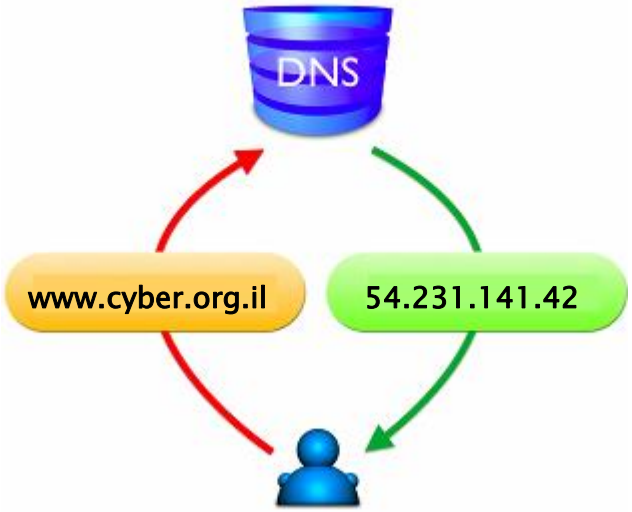
- The purpose of the DNS protocol
- Domain name hierarchy
  - Root
  - TLD
  - Zone
- Query types
  - A, AAAA, PTR, CNAME
  - Iterative, recursive
- Hand On
  - Analysis of DNS packets using Wireshark
  - Creation of DNS packets using Scapy

Domain Names

- Why are domain names even required?

DNS Goals

- DNS - Domain Name System
- DNS is an application layer protocol
- Maps domain name to IP addresses
- Without DNS, it is impossible to maintain the Internet
- Why?



Imagine the world without DNS

- Storage of domain-IP records on every device
  - Storage Volume
  - Keeping the store updated

- Hands on
  - C:\Windows\system32\drivers\etc\hosts

Imagine a Single Global DNS Server

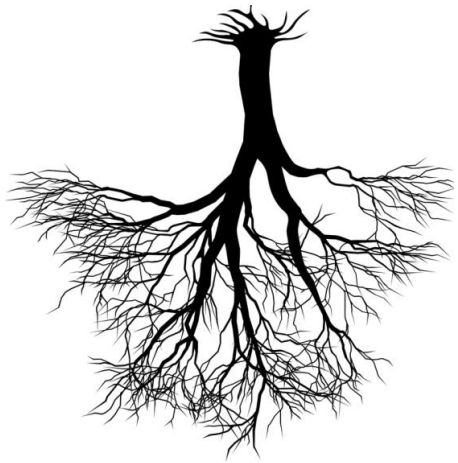
- One server, holding all DNS records
- What are the problems?

Imagine a Single Global DNS Server

- One server, holding all DNS records
- What are the problems?
  - Single point of failure
  - Volume of records
  - Search / response times

Domain Names Hierarchy

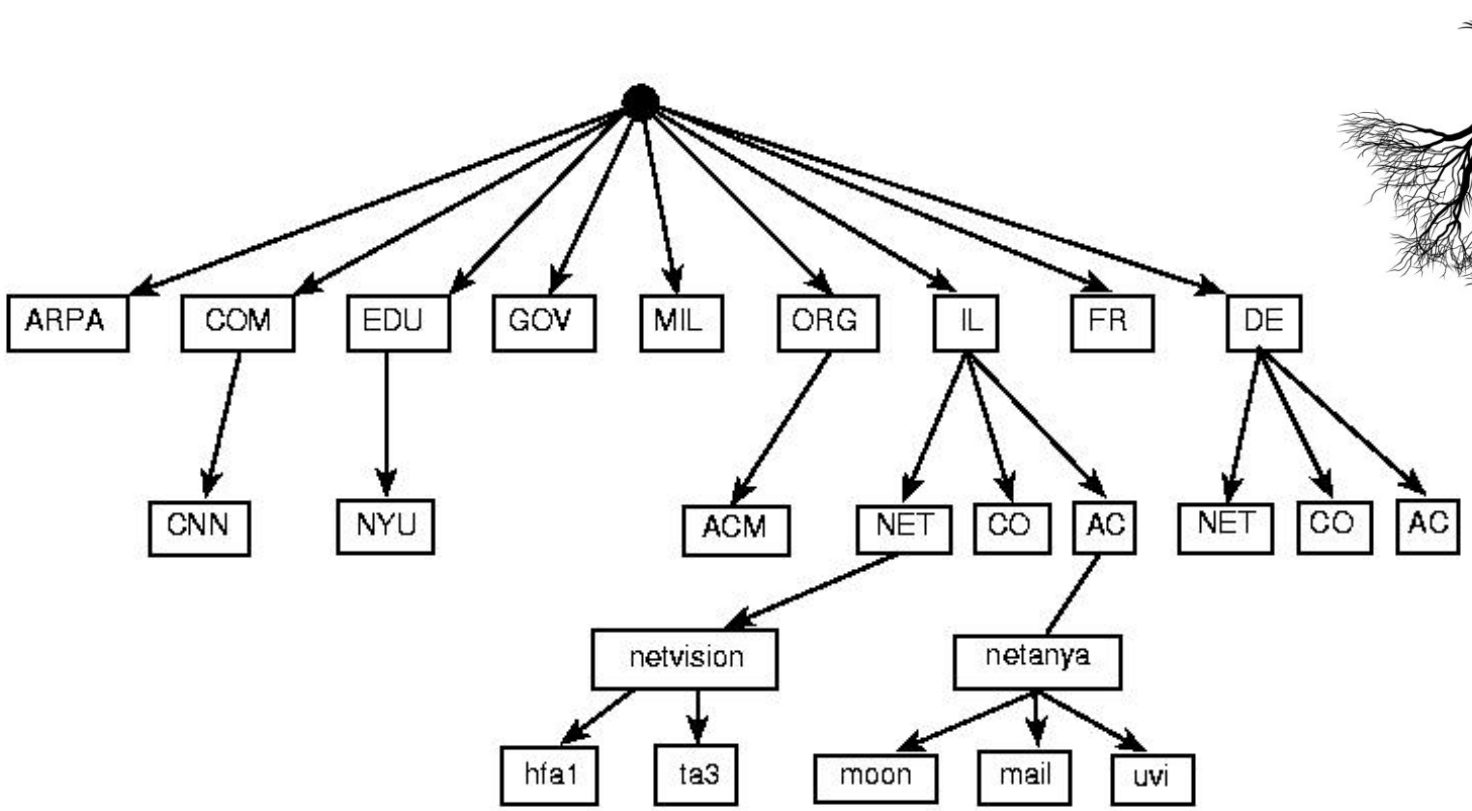
- Principles
  - Multiple servers
  - Each server responsible for a subset of domain names
  - Redundancy
- Domain names are set to levels
- Each DNS server knows only the IPs of domain names in its own level
- “Upside down tree”



Domain Names Hierarchy – cont.

- **Root server** is the base:
  - Address is "."
  - Belongs to ICANN – Internet Corporation of Assigned Numbers and Names
  - There are few https://www.iana.org/domains/root/servers
- **TLD** – Top Level domain:
  - Country codes – il, us, uk, ru …
  - Generic – com, gov, edu, org, net …
- Zones
- DNS server which "owns" a domain name

Domain Names Hierarchy – Example

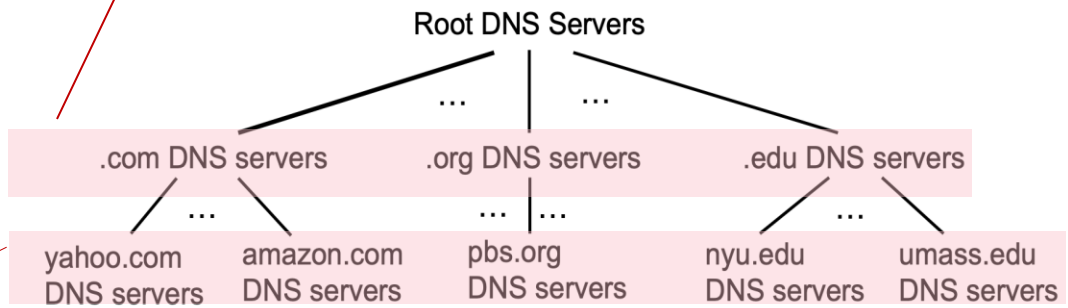


Source :http://mars.netanya.ac.il/~unesco/cdrom/booklet/HTML/NETWORKING/node100.html

Top-Level Domain, and Authoritative Servers

Top-Level Domain (TLD) servers:

- Responsible for .com, .org, .net, .edu, .aero, .jobs, .museums, and all top-level country domains, e.g.: .cn, .uk, .fr, .ca, .jp
- Network Solutions: authoritative registry for .com, .net TLD
- Educause: .edu TLD



Authoritative DNS servers:

- Organization's own DNS server(s), providing authoritative hostname to IP mappings for the organization's named hosts
- Maintained by organization or service provider

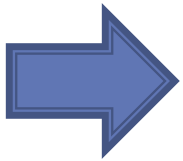
Question

- Do these domains map to the same IP address?
  - www.cyber.org.il
  - www.cyber.il.org

Domain Names Hierarchy – cont.

- Hierarchy is right to left
  - Root, ".", is not written
  - Dot separates between levels
  - Left- the type of service (default is www)

www.google.co.il



root
il
co
google

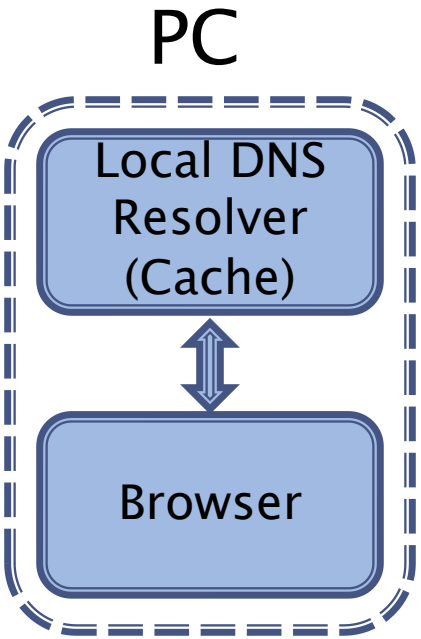
Domain Names Hierarchy – cont.

- Root knows IPs of TLDs
- A TLD server knows only IPs of one level below
  - Ex: www.cyber.org.il, the “il” DNS server knows org.il but not cyber.org.il
- Advantage: A server manages a short list of IPs
  - Simple search
  - Less updates
- A protocol is required to find IPs

DNS Protocol Operation

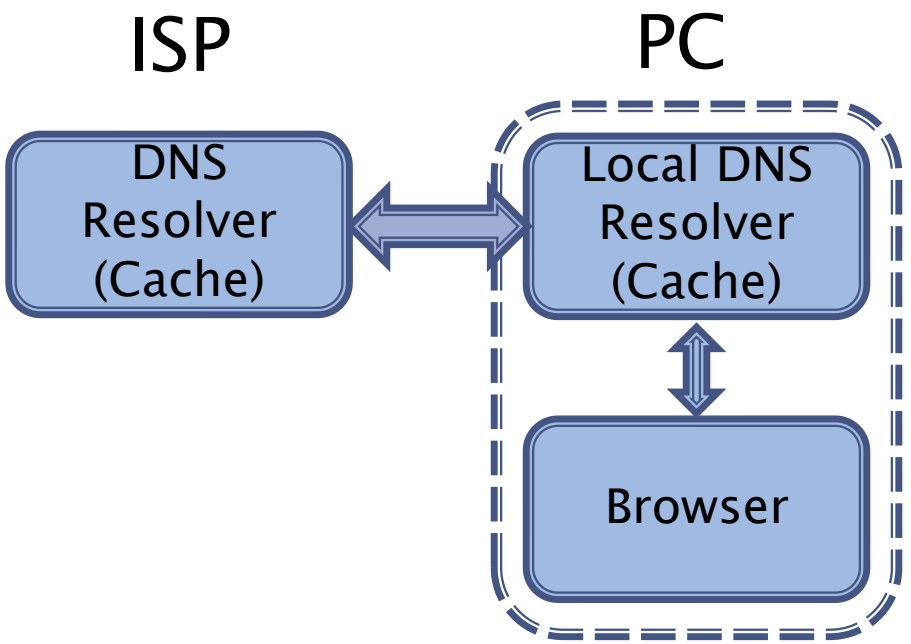
Stage 1: Browser requests domain name

- Operating system checks if IP is in the cache
- If yes – return IP address(es)



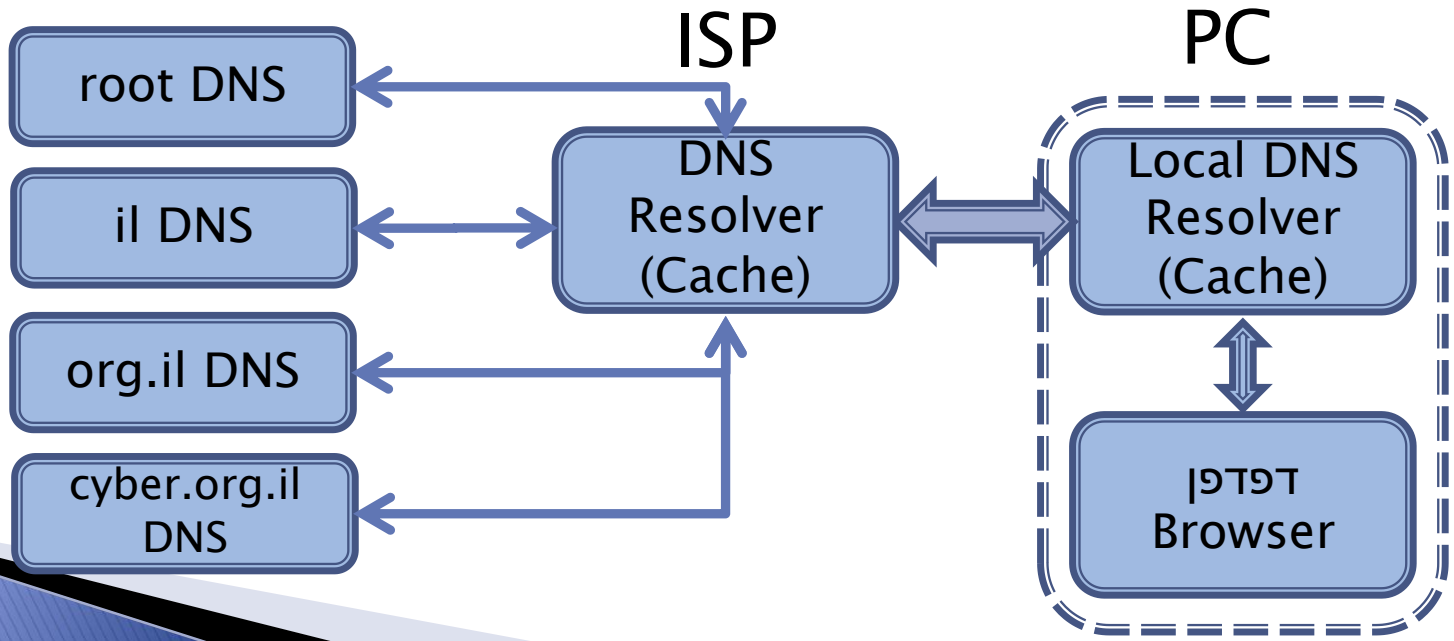
DNS Protocol Operation – cont.

- **Stage 2**: PC makes DNS request to server
  - Typically, ISP server (defined in browser)
  - Named "DNS resolver"
  - Has cache
  - If IP found – return IP address(es)



DNS Protocol Operation – cont.

- **Stage 3:** DNS resolver seeks IP using other DNS servers
- For example, for www.cyber.org.il:
  - From root, requests IP address of “il” DNS server
  - From “il” DNS server, requests IP address of “org.il” DNS server
  - From “org.il” DNS server, requests IP address of “cyber.org.il” DNS server



DNS Query Types

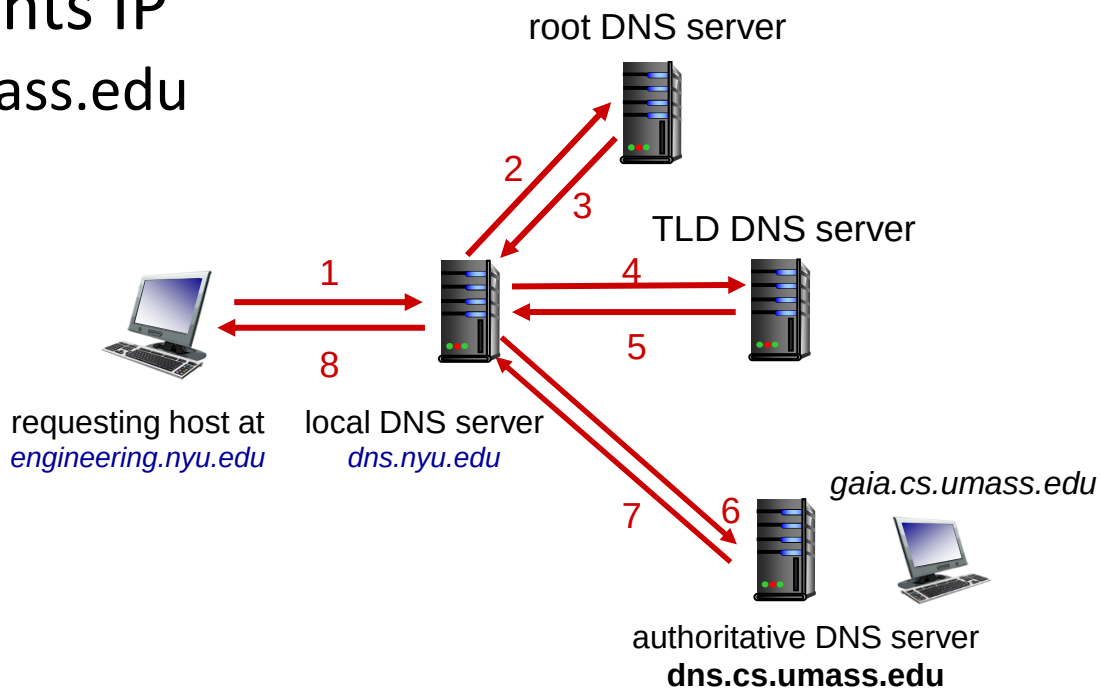
- The DNS resolver returns final IP address(es)
  - Iterative query will return “full service”
- The other servers return only next server’s IP
  - Recursive query
- Protocol field: Recursion Desired (RD flag)
- Perform ex. 4.14 and look for the flag

DNS name resolution: iterative query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterative query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

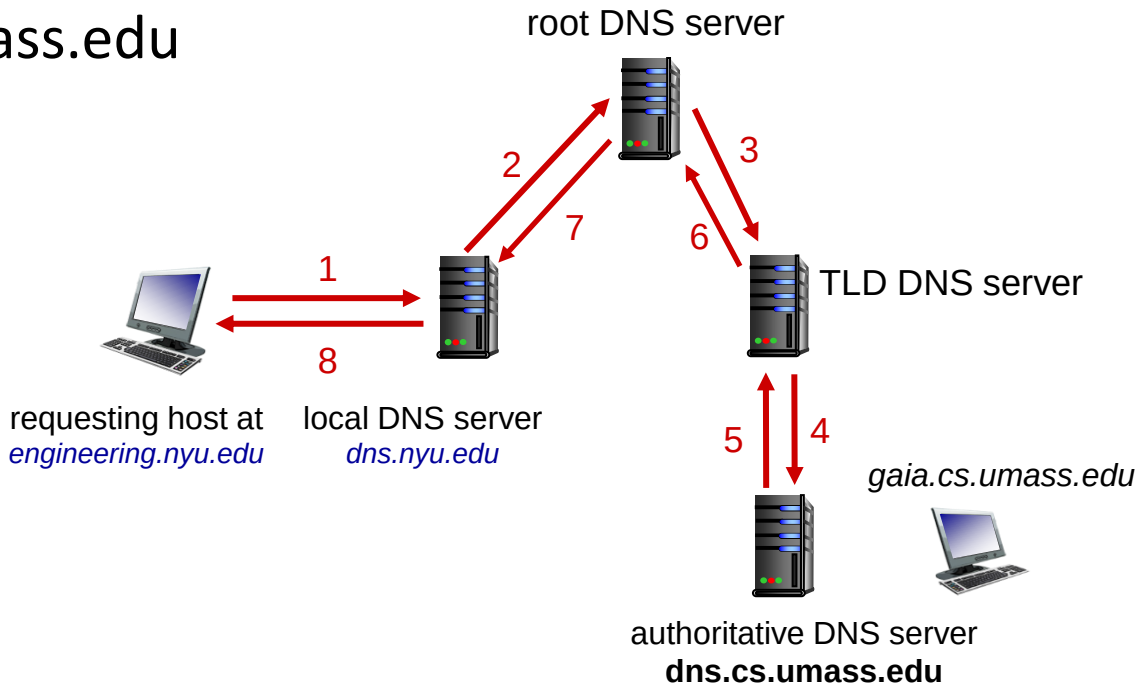


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS Iterative Queries

- Demo – how to find:
  - www.jct.ac.il
  - www.facebook.com
- Hands on – find:
  - www.palmach.org.il
  - www.amazon.com

Reverse Mapping

- Query type:
  - Type ‘A’: Domain -> IP
  - Type ‘PTR’: IP -> Domain
- Perform ex 4.15
  - What is the domain name of IP 8.8.8.8?

DNS records

DNS: distributed database storing resource records (RR)

RR format: (`name, value, type, ttl`)

type=A

- `name` is hostname
- `value` is IP address

type=NS

- `name` is domain (e.g., foo.com)
- `value` is hostname of authoritative name server for this domain

type=CNAME

- `name` is alias name for some "canonical" (the real) name
- www.ibm.com is really servereast.backup2.ibm.com (canonical name)
- `value` is canonical name

type=MX

- `value` is name of SMTP mail server associated with `name`

TTL– Time To Live

- How can the DNS resolver in our PC / ISP tell if DNS cache is updated?
  - DNS response has TTL field
  - Perform ex. 4.16, find TTL field in DNS response

Getting your info into the DNS

Example: new startup "Network Utopia"

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com

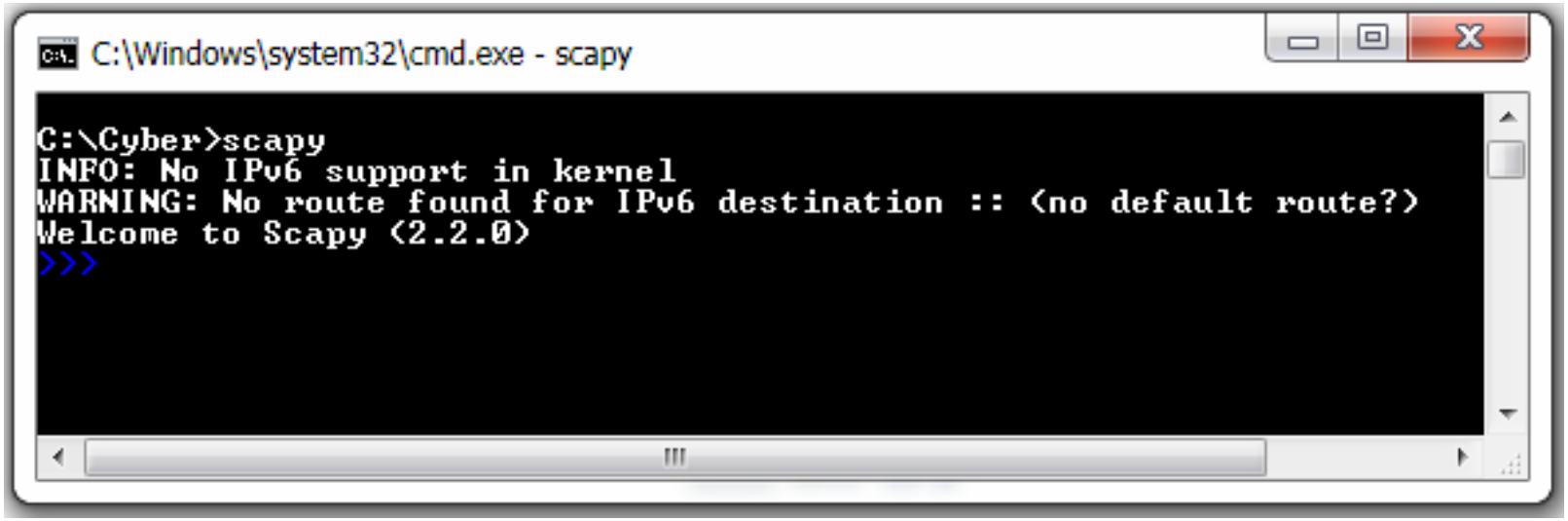

Scapy

Intro

- We learned to program sockets
  - Only application layer
- Scapy:
  - Python import library
  - Sniff packets
  - Craft packets

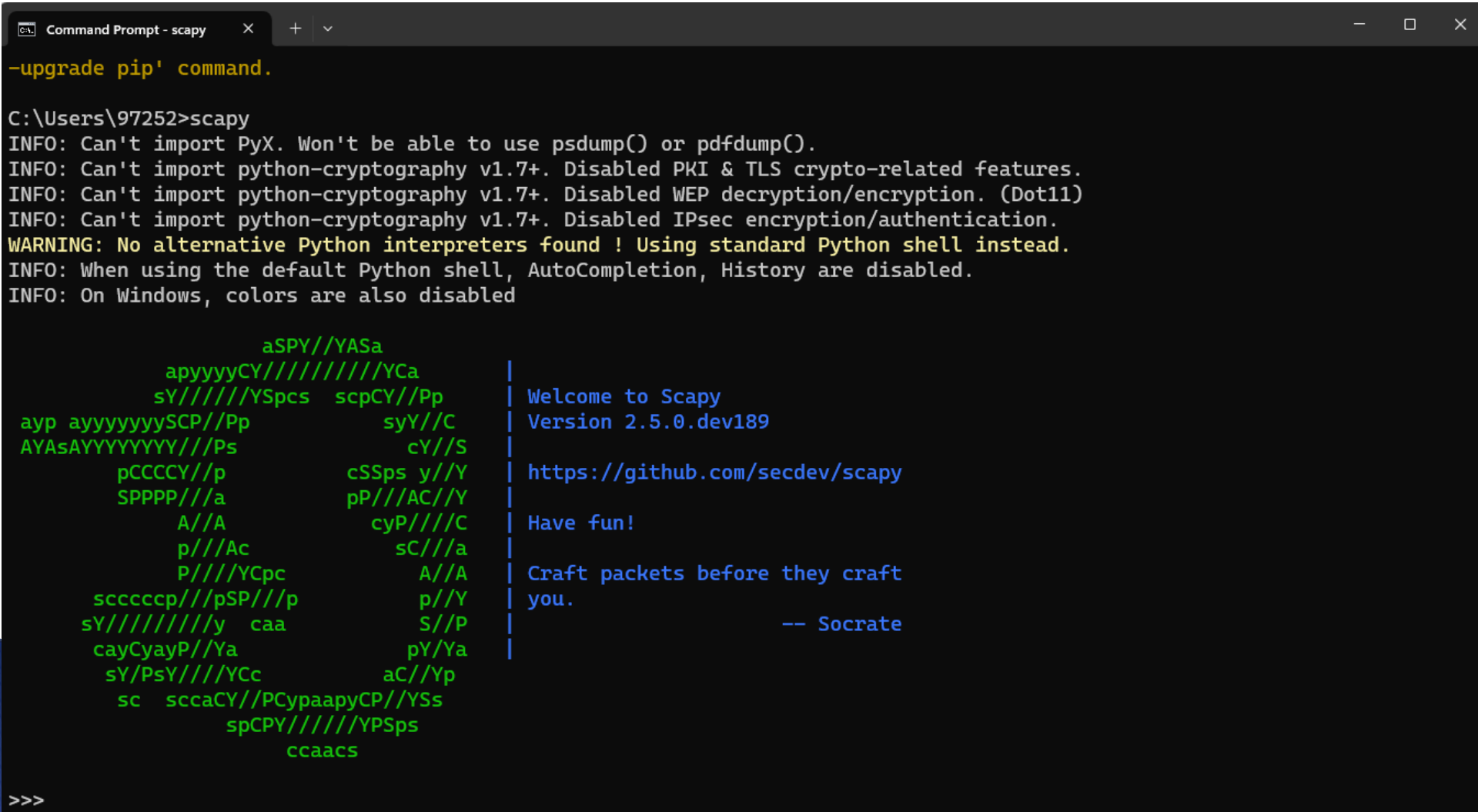
Scapy Fire Up

- CMD -> “scapy”



Scapy Fire Up

- CMD -> “scapy”



Function Sniff

- Let's sniff packets

```
>>> packets = sniff(count=2)
>>> packets
<Sniffed: TCP:1 UDP:1 ICMP:0 Other:0>
>>> packets.summary()
Ether / IPv6 / UDP fe80::d0d7:e117:159e:a608:59302 > ff02::c:ssdp / Raw
Ether / IP / TCP 10.0.0.2:61649 > 173.194.65.188:5228 A / Raw
>>>
```

- Packets are stored in list
- Access same as in python

```
>>> packets[1]
<Ether  dst=c4:12:f5:f8:ab:3e src=38:60:77:25:8e:19 type=0x800 |<IP  version=4L
ihl=5L tos=0x0 len=41 id=15749 flags=DF frag=0L ttl=128 proto=tcp chksum=0x0 src
=10.0.0.2 dst=173.194.65.188 options=[] |<TCP  sport=61649 dport=5228 seq=875105
099 ack=2985448462L dataofs=5L reserved=0L flags=A window=16412 chksum=0xf99b ur
gptr=0 options=[] |<Raw  load='\x00' |>>>>
>>>
```

Function Sniff

- Let's sniff packets

```
>>> packets = sniff(count=2)
>>> packets
<Sniffed: TCP:2 UDP:0 ICMP:0 Other:0>
>>> packets.summary()
Ether / IP / TCP 192.168.1.123:50390 > 172.217.22.37:https A / Raw
Ether / IP / TCP 172.217.22.37:https > 192.168.1.123:50390 A
>>>
```

- Packets are stored in list
- Access same as in python

```
>>> packets[1]
<Ether  dst=04:6c:59:cb:2c:f8 src=d4:35:1d:5d:0c:49 type=IPv4 |<IP  version=4 ihl=5 tos=0x0 len=52 id=41042 flags=DF frag=0 ttl=70 proto=tcp chksum=0xf50
 src=172.217.22.37 dst=192.168.1.123 |<TCP  sport=https dport=50390 seq=1556080982 ack=2127713084 dataofs=8 reserved=0 flags=A window=83 chksum=0x7175 ur
gptr=0 options=[('NOP', None), ('NOP', None), ('SAck', (2127713083, 2127713084))] |>>>
>>> packets[0]
<Ether  dst=d4:35:1d:5d:0c:49 src=04:6c:59:cb:2c:f8 type=IPv4 |<IP  version=4 ihl=5 tos=0x0 len=41 id=54350 flags=DF frag=0 ttl=128 proto=tcp chksum=0x0
src=192.168.1.123 dst=172.217.22.37 |<TCP  sport=50390 dport=https seq=2127713083 ack=1556080982 dataofs=5 reserved=0 flags=A window=516 chksum=0x853d ur
gptr=0 |<Raw  load=b'\x00' |>>>>
>>>
```

Filter DNS packets (ex 5.1)

- Scapy identifies common protocols
- Custom filtering- use lfilter (small "L")
- Example- DNS filtering

```
>>> def filter_dns(packet):
...     return DNS in packet
>>> packets = sniff(count=4, lfilter=filter_dns)
>>> packets.summary()
Ether / IP / UDP / DNS Qry "138.0.0.10.in-addr.arpa."
Ether / IP / UDP / DNS Ans "Broadcom.Home."
Ether / IP / UDP / DNS Qry "www.google.com.Home."
Ether / IP / UDP / DNS Ans
```

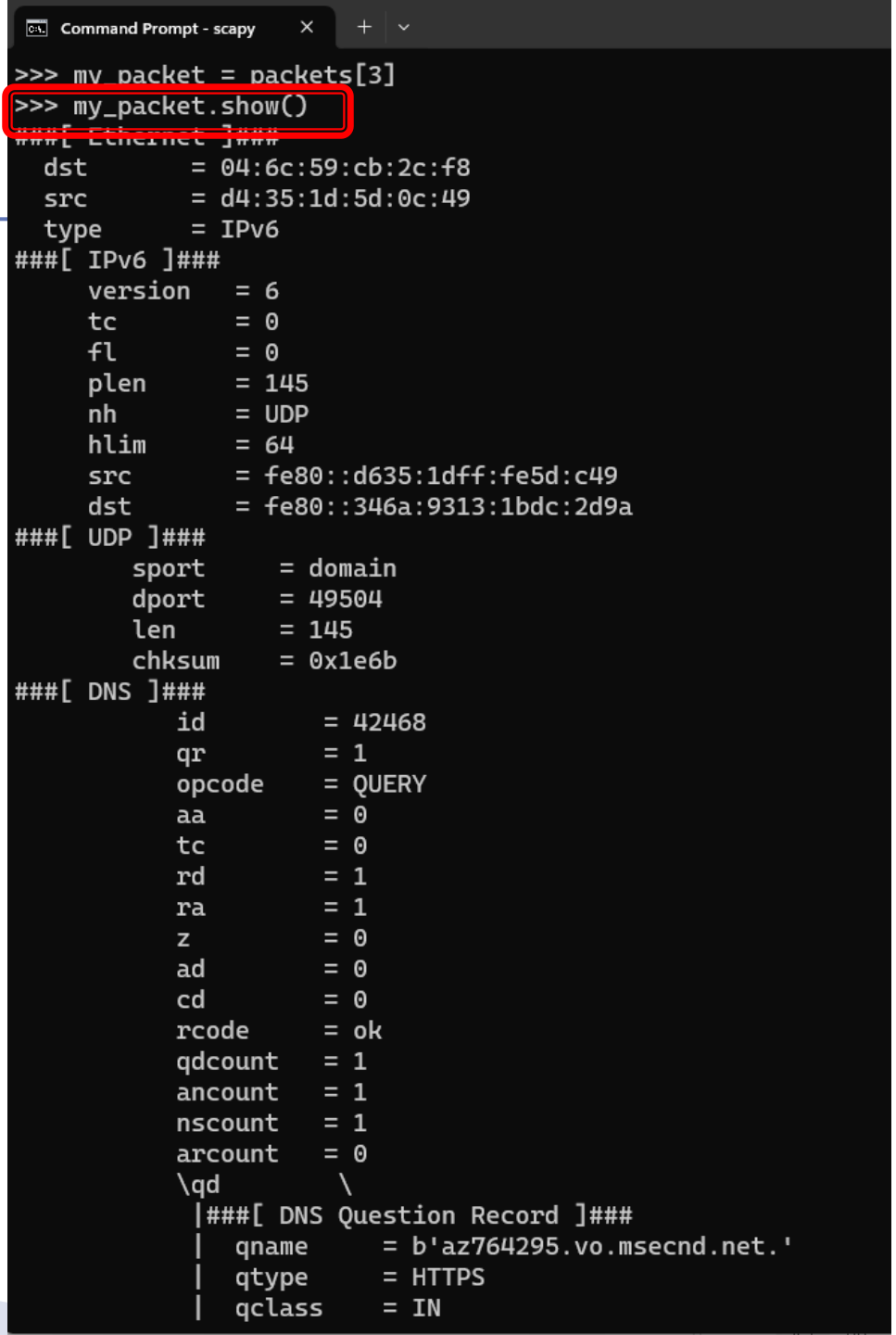
Filter DNS packets (ex 5.1)

- Scapy identifies common protocols
- Custom filtering- use lfilter (small "L")
- Example- DNS filtering

```
>>> def filter_dns(packet):
...  return DNS in packet
...
>>> packets = sniff(count=4, lfilter=filter_dns)
>>> packets.summary()
Ether / IPv6 / UDP / DNS Qry b'az764295.vo.msecnd.net.'
Ether / IPv6 / UDP / DNS Qry b'az764295.vo.msecnd.net.'
Ether / IPv6 / UDP / DNS Ans b'cs22.wpc.v0cdn.net.'
Ether / IPv6 / UDP / DNS Ans b'cs22.wpc.v0cdn.net.'
>>>
```

Show()

- Nice presentation of packet's fields



Checking Field Values

- "QUERY" is a constant value, defined in DNS protocol

```
>>> my_packet[DNS].opcode
0L
```

"L" indicates long integer

- Conclusion: 0 means "Query"

```
Command Prompt - scapy
>>> my_packet[DNS].show()
###[ DNS ]###
  id        = 42468
  qr        = 1
  opcode    = QUERY
  aa        = 0
  tc        = 0
  rd        = 1
  ra        = 1
  z         = 0
  ad        = 0
  cd        = 0
  rcode     = ok
  qdcount   = 1
  ancount   = 1
  nscount   = 1
  arcount   = 0
  \qd        \
   |###[ DNS Question Record ]###
   |  qname     = b'az764295.vo.msecnd.net.'
   |  qtype     = HTTPS
   |  qclass    = IN
  \an        \
   |###[ DNS Resource Record ]###
   |  rrname    = b'az764295.vo.msecnd.net.'
   |  type      = CNAME
   |  rclass    = IN
   |  ttl       = 1532
   |  rdlen     = None
   |  rdata     = b'cs22.wpc.v0cdn.net.'
  \ns        \
   |###[ DNS SOA Resource Record ]###
   |  rrname    = b'wpc.v0cdn.net.'
   |  type      = SOA
   |  rclass    = IN
   |  ttl       = 569
   |  rdlen     = None
   |  mname     = b'ns1.v0cdn.net.'
   |  rname     = b'noc.edgecast.com.'
   |  serial    = 1609952856
   |  refresh   = 3600
   |  retry     = 600
```

Filter DNS Packets

- Suppose we need to filter DNS queries of type CNAME:
  - DNS – learned
  - Query – learned
  - Type CNAME:

```
>>> my_packet[DNSQR].qtype
65
```

Command Prompt - scapy

```
>>> my_packet[DNS].show()
###[ DNS ]###
  id        = 42468
  qr        = 1
  opcode    = QUERY
  aa        = 0
  tc        = 0
  rd        = 1
  ra        = 1
  z         = 0
  ad        = 0
  cd        = 0
  rcode     = ok
  qdcount   = 1
  ancount   = 1
  nscount   = 1
  arcount   = 0
  \qd        \
   |###[ DNS Question Record ]###
   |  qname     = b'az764295.vo.msecnd.net.'
   |  qtype     = HTTPS
   |  qclass    = IN
  \an        \
   |###[ DNS Resource Record ]###
   |  rrname    = b'az764295.vo.msecnd.net.'
   |  type      = CNAME
   |  rclass    = IN
   |  ttl       = 1532
   |  rdlen     = None
   |  rdata     = b'cs22.wpc.v0cdn.net.'
  \ns        \
   |###[ DNS SOA Resource Record ]###
   |  rrname    = b'wpc.v0cdn.net.'
   |  type      = SOA
   |  rclass    = IN
   |  ttl       = 569
   |  rdlen     = None
   |  mname     = b'ns1.v0cdn.net.'
   |  rname     = b'noc.edgecast.com.'
   |  serial    = 1609952856
   |  refresh   = 3600
   |  retry     = 600
```

Filter DNS packets

- Suppose we need to filter DNS queries of type A:
  - DNS – learned
  - Query – learned
  - Type A:

```
            |  qname     = b'az7
            |  qtype     = A
            |  qclass    = IN
          \an        \
          \ns        \
          \ar        \

>>> my_packet[DNSQR].qtype
1
```

```
>>> my_packet = packets[0]
>>> my_packet.summary()
"Ether / IPv6 / UDP / DNS Qry b'az764295.vo.msecnd.net.'"
>>> my_packet.show()
###[ Ethernet ]###
  dst       = d4:35:1d:5d:0c:49
  src       = 04:6c:59:cb:2c:f8
  type      = IPv6
###[ IPv6 ]###
     version   = 6
     tc        = 0
     fl        = 1035051
     plen      = 48
     nh        = UDP
     hlim      = 64
     src       = fe80::346a:9313:1bdc:2d9a
     dst       = fe80::d635:1dff:fe5d:c49
###[ UDP ]###
        sport     = 58742
        dport     = domain
        len       = 48
        chksum    = 0xd13
###[ DNS ]###
           id        = 59364
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 0
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = b'az764295.vo.msecnd.net.'
```

Filter DNS Packets

```
>>> my_packet[DNS].show()
###[ DNS ]###
  id= 4
  qr= 0L
  opcode= QUERY
  aa= 0L
  tc= 0L
  rd= 1L
  ra= 0L
  z= 0L
  rcode= ok
  qdcount= 1
  ancount= 0
  nscount= 0
  arcount= 0
  \qd\
   |###[ DNS Question Record ]###
   |  qname= 'www.google.com.'
   |  qtype= A
   |  qclass= IN
  an= None
  ns= None
  ar= None
>>>
```

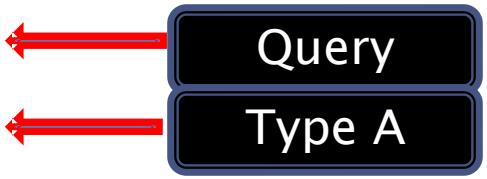
- Suppose we need to filter DNS queries of type A:
  - DNS – learned
  - Query – learned
  - Type A:

```
>>> my_packet[DNSQR].qtype
1
```

Filter DNS Packets – cont.

- Refine our filter:

```
>>> def filter_dns(packet):
        if (DNS in packet) and (DNSQR in packet):
            return (packet[DNS].opcode==0)
                and (packet[DNSQR].qtype==1)
```



Processing Post Filtering

- Scapy enables processing of filtered packets
- Example- print domain names of DNS queries

```
>>> def print_query_name(dns_packet):
        print(dns_packet[DNSQR].qname)
```

- Use “prn” optional parameter

PRN Parameter

```
>>> sniff(count=10, lfilter=filter_dns, prn=print_query_name)
www.google.co.il.
www.google.co.il.
www.google.com.
www.google.com.
apis.google.com.
lh4.googleusercontent.com.
plus.google.com.
apis.google.com.
lh4.googleusercontent.com.
plus.google.com.
<Sniffed: TCP:0 UDP:10 ICMP:0 Other:0>
>>>
```

- If no results, flush DNS cache
  - ipconfig/flushdns

Scapy from Python Script

```
import sys
i, o, e = sys.stdin, sys.stdout, sys.stderr
from scapy.all import *
sys.stdin, sys.stdout, sys.stderr = i, o, e
```

Exercise – Filter HTTP packets

- Perform ex 5.2
- Note, for python 3 “decode” method required:

```
return packet[Raw].load.decode().startswith('GET')
```

Interim Summary

- We learned how to use Scapy to sniff packets
- We learned how to process post-filtering
- How about crafting packets?

Nslookup by Scapy

- Nslookup tool –

```
c:\Cyber>nslookup www.google.com
Server:  Broadcom.Home
Address:  10.0.0.138

Non-authoritative answer:
Name:    www.google.com
Addresses:  2a00:1450:4009:800::2004
          216.58.208.36
```

- We shall soon write our own ☺

```
c:\Cyber>python my_DNS.py
Please enter domain address
www.google.com
Begin emission:
Finished to send 1 packets.
...*
Received 4 packets, got 1 answers, remaining 0 packets
IP address number 1 is: 216.58.208.68
```

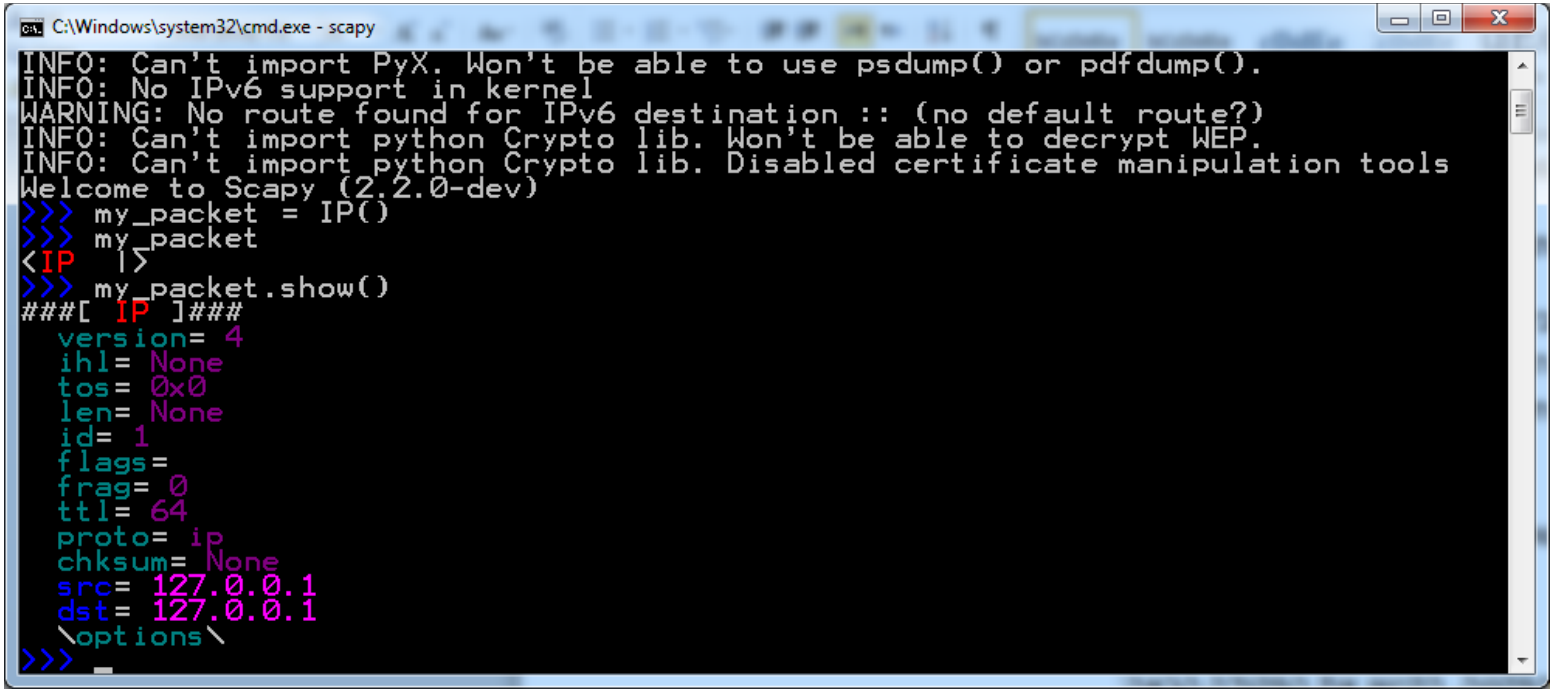
Nslookup by Scapy

- “Tailor made” packets
- Learning phases:
  - Create packets
  - Add layers
  - Set values
  - Send and receive responses
  - Use in python script

Create “Skeleton” Packet

```
>>> my_packet = IP()
>>> my_packet.show()
```

- Network layer only, IP protocol, default values



Set Parameters

- Set destination:

  >>> my_packet.dst = '10.1.1.1'

```
>>> my_packet.dst='10.1.1.1'
>>> my_packet
<IP  dst=10.1.1.1 |>
```

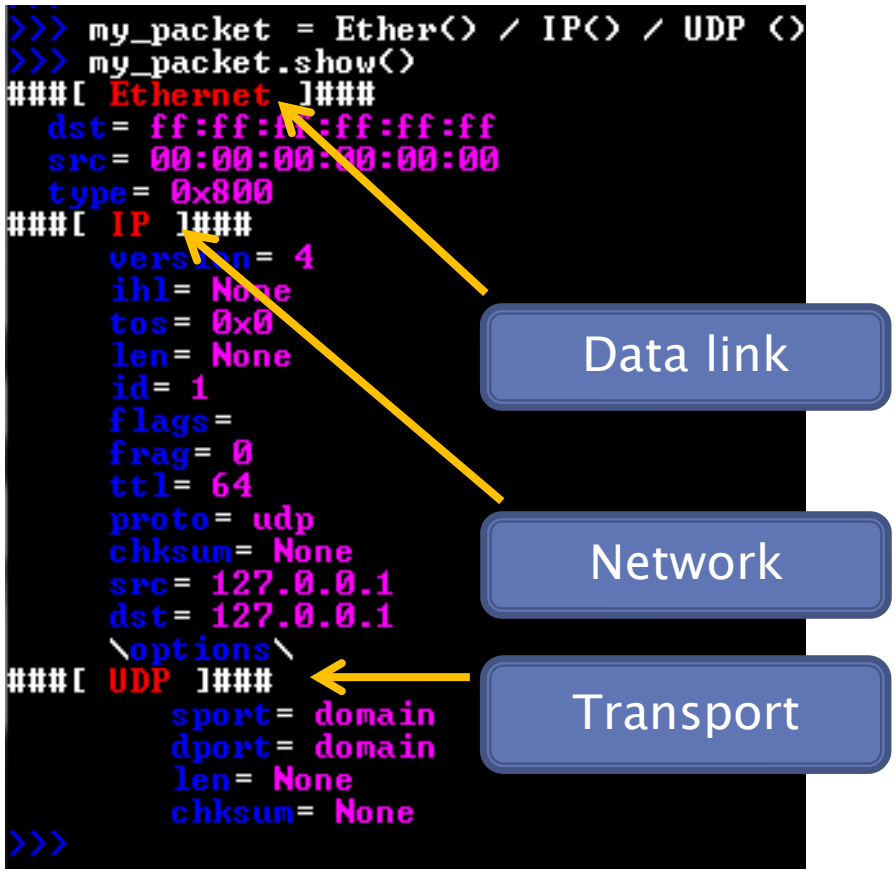
- Other option:

```
>>> my_packet = IP(dst = '10.1.1.2', ttl = 6)
>>> my_packet
<IP  ttl=6 dst=10.1.1.2 |>
```

Adding Layers

- Simply write protocol name
- Order of writing- left to right

```
>>> my_packet = Ether () /
  IP() / UDP()
```



Add Load to Packet

- Let's add HTTP data over TCP packet:

>>> my_packet = Ether() / IP() / TCP() / "GET / HTTP/1.0\r\n\r\n"

```
>>> my_packet = Ether() / IP() / TCP() / Raw("GET / HTTP/1.0\r\n\r\n")
>>> my_packet
<Ether  type=0x800 |<IP  frag=0 proto=tcp |<TCP  |<Raw  load='GET / HTTP/1.0\r\n
\r\n' |>>>>
```

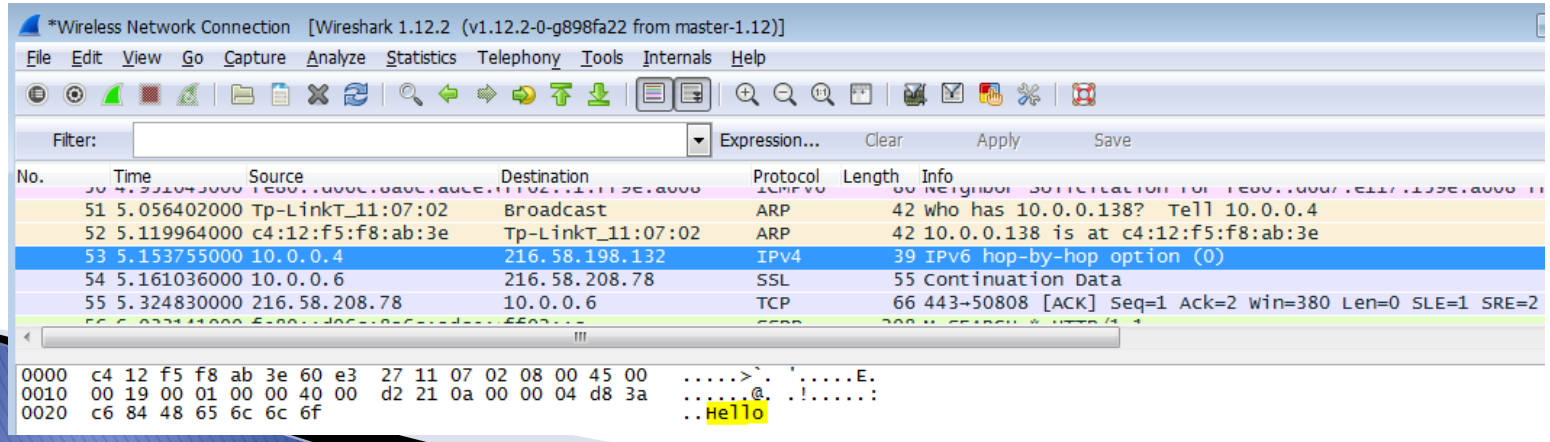
Sending Packets

- Scapy can translate domain name to IP
- Let's load some data and send it to Google:

  >>> my_packet = IP(dst ='www.google.com') / 'Hello'

  >>> send(my_packet)

  - Use Wireshark and find your packet



Nslookup by Scapy

- Self study ex. 6.10, 6.11
- Craft a DNS packet:
  - DNS server's IP
  - Destination port
  - Flags
  - Requested domain name
  - Query type
- From response packet, extract responses
- Important:
  - More than one response is possible (www.youtube.com)
  - How can one find how many answers exist?
  - Extract only answers of given type

http://itgeekchronicles.co.uk/2014/05/12/scapy-iterating-over-dns-responses/