Scapy, a packet manipulation tool

Guillaume Valadon

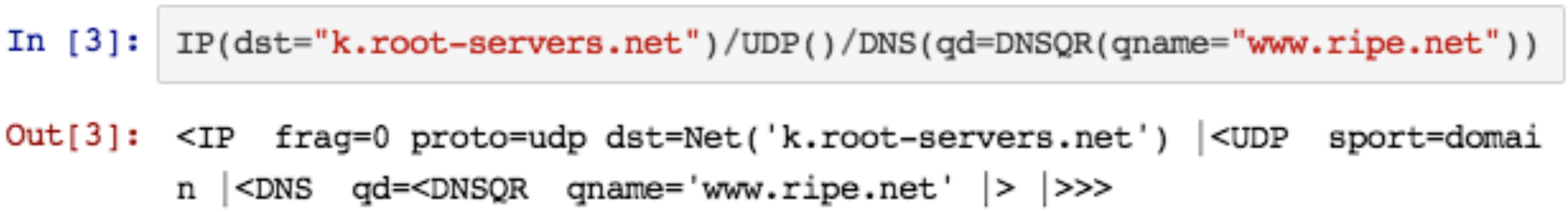
RIPE 70 - May, 14 2015

What is Scapy ?

- fast packet manipulation in Python
  - send, receive, inject, save, modify, ...
- default values that work
- hidden tricks: checksum computations, interface selection, ...
- developped by Philippe Biondi since 2003
- maintained by Pierre Lalet and Guillaume Valadon since 2013

Scapy as a command line tool

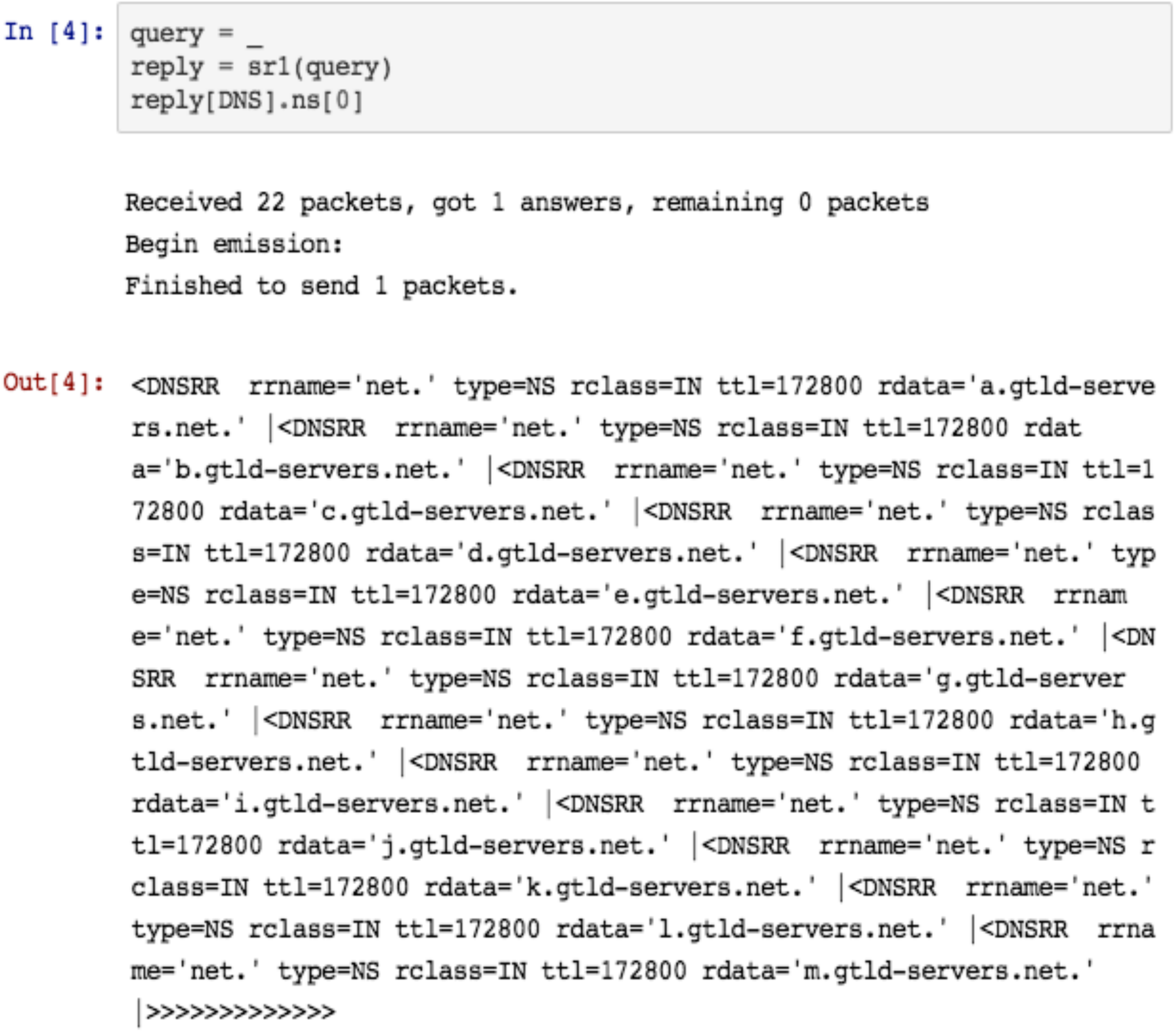
Packet built layer by layer (Ether, IP, TCP, ...) using the slash operator, such as:



```
In [3]:  IP(dst="k.root-servers.net")/UDP()/DNS(qd=DNSQR(qname="www.ripe.net"))

Out[3]:  <IP  frag=0 proto=udp dst=Net('k.root-servers.net') |<UDP  sport=domai
         n |<DNS  qd=<DNSQR  qname='www.ripe.net' |> |>>>
```

Scapy matches queries and replies:



```
In [4]:  query = _
         reply = sr1(query)
         reply[DNS].ns[0]
```

```
Received 22 packets, got 1 answers, remaining 0 packets
Begin emission:
Finished to send 1 packets.
```

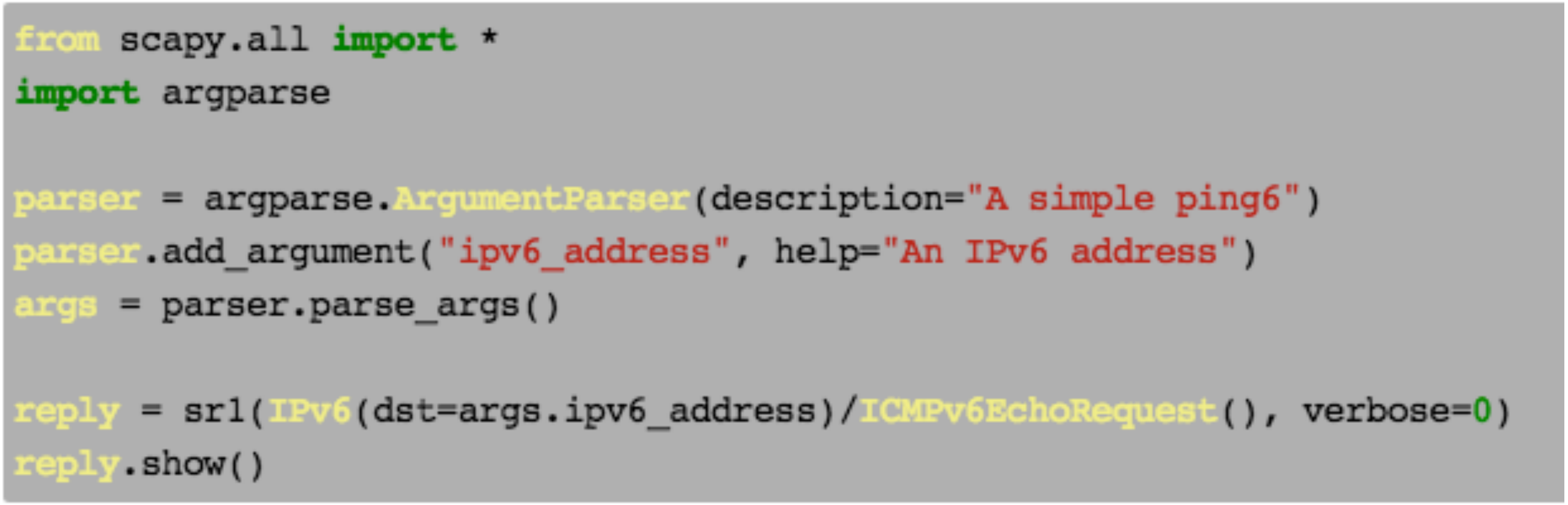
```
Out[4]:  <DNSRR  rrname='net.' type=NS rclass=IN ttl=172800 rdata='a.gtld-serve
         rs.net.' |<DNSRR  rrname='net.' type=NS rclass=IN ttl=172800 rdat
         a='b.gtld-servers.net.' |<DNSRR  rrname='net.' type=NS rclass=IN ttl=1
         72800 rdata='c.gtld-servers.net.' |<DNSRR  rrname='net.' type=NS rclas
         s=IN ttl=172800 rdata='d.gtld-servers.net.' |<DNSRR  rrname='net.' typ
         e=NS rclass=IN ttl=172800 rdata='e.gtld-servers.net.' |<DNSRR  rrnam
         e='net.' type=NS rclass=IN ttl=172800 rdata='f.gtld-servers.net.' |<DN
         SRR  rrname='net.' type=NS rclass=IN ttl=172800 rdata='g.gtld-server
         s.net.' |<DNSRR  rrname='net.' type=NS rclass=IN ttl=172800 rdata='h.g
         tld-servers.net.' |<DNSRR  rrname='net.' type=NS rclass=IN ttl=172800
         rdata='i.gtld-servers.net.' |<DNSRR  rrname='net.' type=NS rclass=IN t
         tl=172800 rdata='j.gtld-servers.net.' |<DNSRR  rrname='net.' type=NS r
         class=IN ttl=172800 rdata='k.gtld-servers.net.' |<DNSRR  rrname='net.'
         type=NS rclass=IN ttl=172800 rdata='l.gtld-servers.net.' |<DNSRR  rrna
         me='net.' type=NS rclass=IN ttl=172800 rdata='m.gtld-servers.net.'
         |>>>>>>>>>>>>>
```

Some useful functions, for example:

```
In [5]: wrpcap("/tmp/dns.pcap", reply)
```

Scapy as a Python module

A simple ping6 with Scapy:



```
from scapy.all import *
import argparse

parser = argparse.ArgumentParser(description="A simple ping6")
parser.add_argument("ipv6_address", help="An IPv6 address")
args = parser.parse_args()

reply = sr1(IPv6(dst=args.ipv6_address)/ICMPv6EchoRequest(), verbose=0)
reply.show()
```

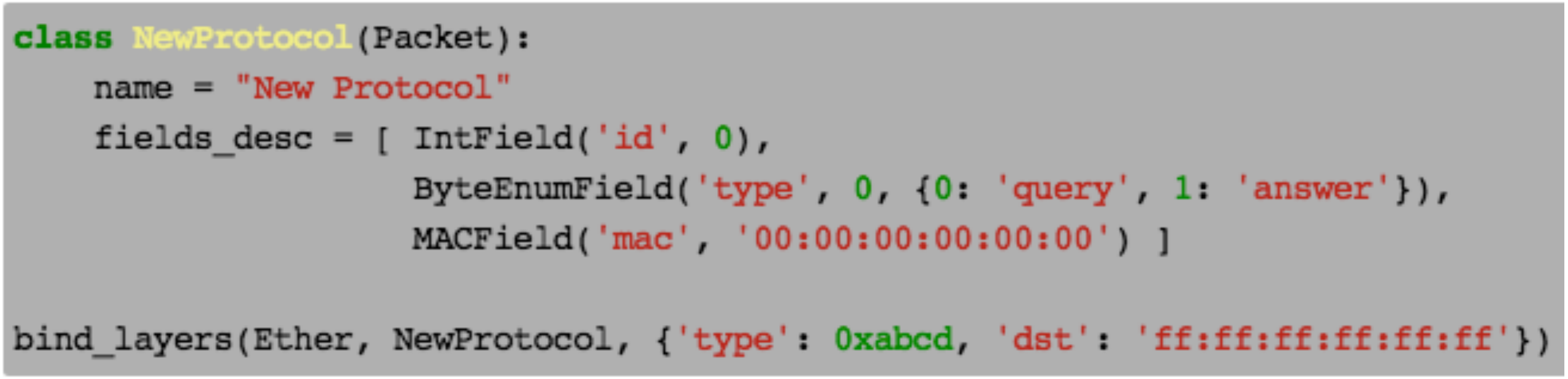
Supported protocols

- IP, IPv6, UDP, TCP, ICMP, ICMPv6, ...
- DNS/DNSSEC, SNMP, DHCP, DHCPv6, HSRP, ...
- RIP, BGP, Mobile IPv6, ...

- contributions: OpenFlow, MPLS, HomePlug AV, ..

Adding a new protocol

Let's add a new protocol on top of Ethernet:



```
class NewProtocol(Packet):
    name = "New Protocol"
    fields_desc = [ IntField('id', 0),
                    ByteEnumField('type', 0, {0: 'query', 1: 'answer'}),
                    MACField('mac', '00:00:00:00:00:00') ]

bind_layers(Ether, NewProtocol, {'type': 0xabcd, 'dst': 'ff:ff:ff:ff:ff:ff'})
```

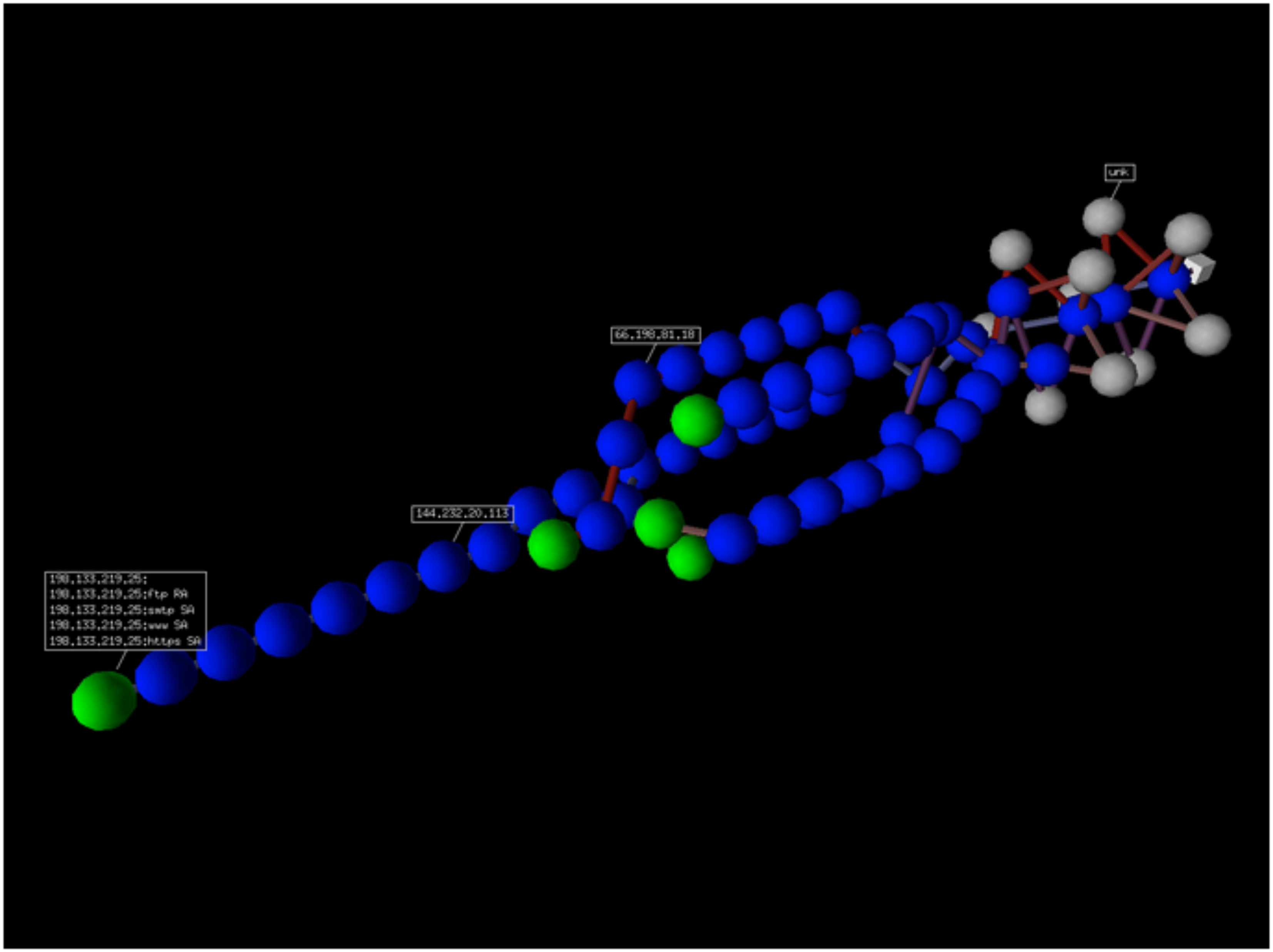
More features are available

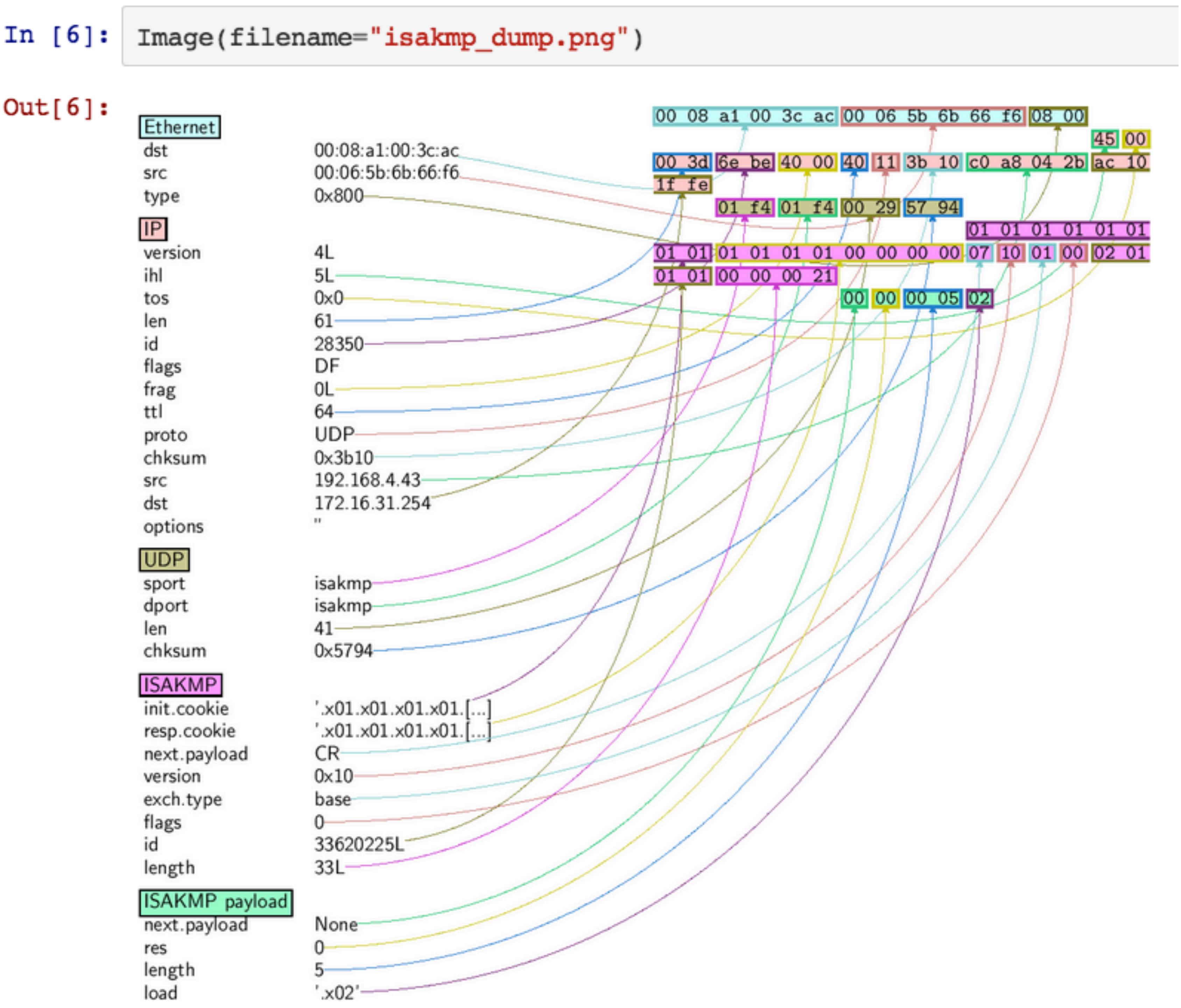
- answering machines
- automation
- …

In [8]:

```
Image(filename="trace3d.png")
```

Out[8]:





```
In [6]: Image(filename="isakmp_dump.png")
```

```
Out[6]:
Ethernet
dst            00:08:a1:00:3c:ac
src            00:06:5b:6b:66:f6
type           0x800
IP
version        4L
ihl            5L
tos            0x0
len            61
id             28350
flags          DF
frag           0L
ttl            64
proto          UDP
chksum         0x3b10
src            192.168.4.43
dst            172.16.31.254
options        ''
UDP
sport          isakmp
dport          isakmp
len            41
chksum         0x5794
ISAKMP
init.cookie    '.x01.x01.x01.x01.[...]
resp.cookie    '.x01.x01.x01.x01.[...]
next.payload   CR
version        0x10
exch.type      base
flags          0
id             33620225L
length         33L
ISAKMP payload
next.payload   None
res            0
length         5
load           '.x02'
```

```
00 08 a1 00 3c ac 00 06 5b 6b 66 f6 08 00
                                       45 00
00 3d 6e be 40 00 40 11 3b 10 c0 a8 04 2b ac 10
1f fe
      01 f4 01 f4 00 29 57 94
                              01 01 01 01 01 01
01 01 01 01 01 01 00 00 00 00 07 10 01 00 02 01
01 01 00 00 00 21
                  00 00 00 05 02
```

Where ?

- Scapy works on Linux, *BSD, MAC OS X
  - the Windows port does not work anymore
- stable version: 2.3.1
  - pip, arch, and gentoo
- development version on bitbucket:
  - hg clone https://bitbucket.org/secdev/scapy/

How can you help ?

- tell that you use Scapy
- report issues on Bitbucket
- share your protocols
- invite use to give tutorials