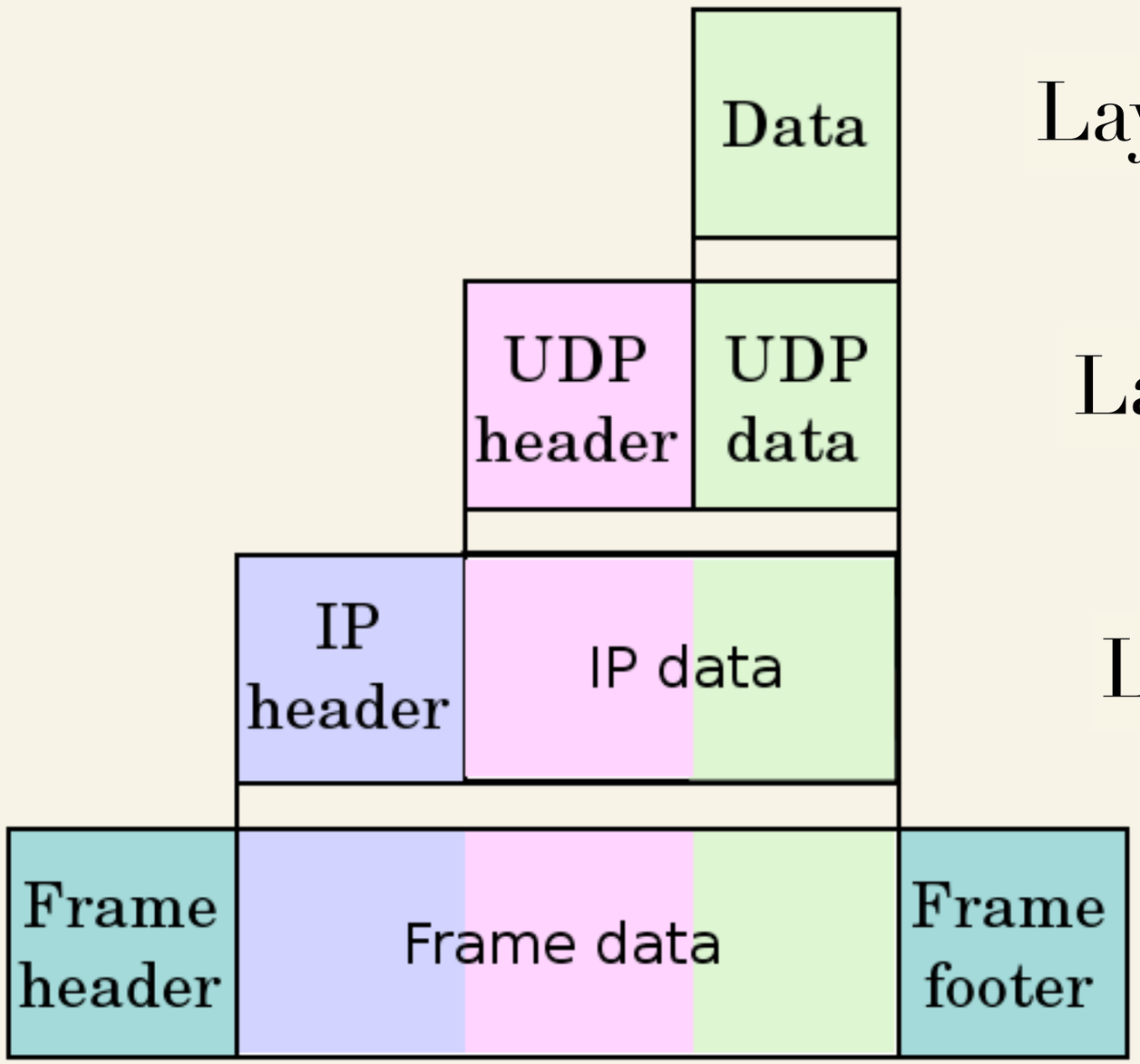
Reverse Engineering and Exploitation

Dr. Carl Pulley
c.j.pulley@hud.ac.uk

TCP/IP Model



Layer 5 (Application)

Layer 4 (Transport)

Layer 3 (Internet)

Layer 2 (Link)

Packet Manipulation

```
pkt = Ether(src="00:de:ad:be:ef:00") / IP(dst="example.com")

<Ether  src=00:de:ad:be:ef:00 type=0x800 |<IP  dst=Net('example.com') |>>

pkt /= TCP(dport=25)
pkt['TCP'].flags = "S"

<Ether  src=00:de:ad:be:ef:00 type=0x800 |<IP  frag=0 proto=tcp dst=Net
('example.com') |<TCP  dport=smtp flags=S |>>>

pkt.payload = "Hello World!"

<Ether  src=00:de:ad:be:ef:00 |<Raw  load='Hello World!' |>>
```


Ether is a defined layer of pkt and has a payload property - hence the overwrite.
The correct way to set the payload is:
pkt /= "Hello World!"

Fake Client

```
ans, unans = sr(packet, timeout=10)

[ report(pkt[0], pkt[1]) for pkt in ans if check(pkt[0], pkt[1]) ]
```


sr1 will send a group of packets and then return on the first reply - this is more useful in scanning exercises.

Note: in the above, pkt is actually a **pair** of matching IP packets! It's also worth noting that you use list comprehension and filtering a lot with the sr command.

Fake Listener

```
def handler(pkt):
  pkt.sprintf(“%IP.src%:%TCP.sport% -> %IP.dst%:%TCP.dport%”)

sniff(iface=”eth0”, prn=handler, filter=”..”)
```


store=0 (sniffed packets **not** saved); store = 1 (sniffed packets saved)
can limit packets sniffed using count

Fake Message Server

```
def server(pkt):
  pkt.sprintf(“%IP.src%:%UDP.sport% -> %IP.dst%:%UDP.dport%”)
  send(reply(pkt))

sniff(iface=”eth0”, prn=server, filter=”..”)
```


If reply = lambda x: x then we have an echo server. Echo servers can be useful when attempting to understand undocumented protocols.

Fake Session Server

```
def synack(syn):
  synack = IP(dest=syn.src)
  synack /= TCP(sport=syn.dport, dport=syn.sport, ack=syn.seq+1)
  synack['TCP'].flags = "SA"
  synack

def server(pkt):
  if pkt.sprintf("%TCP.flags") == "S":
    pkt.sprintf("S: %IP.src%:%TCP.sport% -> %IP.dst%:%TCP.dport%")
    ack = sr1(synack(pkt))
    if pkt.sprintf("%TCP.flags") == "A":
      ack.sprintf("A: %IP.src%:%TCP.sport% -> %IP.dst%:%TCP.dport%")
      return
  pkt.sprintf("%IP.src%:%TCP.sport% -> %IP.dst%:%TCP.dport%")
  pkt.sprintf("    %IP.payload%")

sniff(iface="eth0", prn=server, filter="..")
```


With session oriented connections, we are generally better off using honeypots (eg. mwcollectd) to fake the interactions with our subject.

Gateways

- All traffic that can not be handled by the network node is sent via the gateway
  - handy way of getting all traffic routed via your monitoring point
  - for monitoring point to programatically interact with such traffic, need to do some kernel routing!

IPTable Manipulations

```
iptables -A INPUT -i eth0 -j NFQUEUE

iptables -t nat -A PREROUTING -i eth0-j DNAT --to-destination ..
```

Other Traffic Manipulations

- Fake ARP replies to poison ARP cache
- Fake DNS replies to poison DNS cache
- Fake HTTP replies to poison browser cache

Faking DHCP replies allows one to influence DNS choices and so manipulate traffic by faking DNS replies.
Should also keep in mind NetBIOS/SMB traffic!

References

- TCP/IP Illustrated: Volume 1 by Richard Stevens
- Security Power Tools by Bryan Burns et. al.