You've been (D)DoSed

So what?

The usual disclaimer

- Opinions expressed in the presentation do not necessary represent the stance of my employer.

- Let me know if I speak too fast and if it's difficult to understand me

Agenda

1. Business Side
    1. Motivation
    2. Considerations
    3. Insurance
    4. Preparedness
    5. Monitoring
2. Attack types
    1. Classification type
    2. Attack surface from OSI perspective
    3. Intro to TCP
    4. Life of a socket
    5. Tools
3. Recognition and mitigation

Audience poll

- Tell me who you are

What DDoS is not

- it is not an act of G--
- you can be prepared
- you can have insurance

...so don't panic ;)

...and no you don't need magical powers to deal it

...you justneed to proper training

Motivation

- Financial Gain
  - Competitions
  - Extortion
  - Divert attention
  - Proof of power
- Political statement
  - Hacktivism
  - “I’m a cooler kid than you”
- Attack types
  - TCP data to a listening port
  - Slowris
- Add your own to the list…

Consider this

How is a DDoS different from CNN pointing to your home page?

How is that different from your primary Internet connection goes down or servers crash?

Reactive vs. nonreactive handling?

DDoS absorption == being able to serve more users faster

Change your attitude!

Few words about insurance

- Insurance is money you pay to be protected from something bad if it is ever to happen
- You can be prepared:
  - Incident response plan
  - Tools
  - Gear
  - Partnerships
- …it may not be sufficient – you should have picked the higher premium policy… ☹

In peace time

- Have a Incident response plan
- You should have your monitoring ahead of time
- When do you need to escalate?
  - Why?!?

Monitoring Impact

- The most neglected resource
- No matter how much traffic they throw at you there is no problem until your users start seeing it
- Use internal monitoring
- Use external monitoring services

In the heat of the moment

- What is actually happening? Focus on the facts
- Collect data (from LBs, systems, network graphs, capture traffic)
- Create a response plan!
- EXECUTE IT!
- Observe! (have the metrics improved?)

- Enough business let's get down to business

Attack Types

- Asymmetric
  - DNS queries
  - SYN flood
- Symetric
  - GET flood
- Reflected
  - Smurf/DNS (spoofed source)
- Brute force or logicstate attacks
- Distributed
  - Any of the above (and many more) ;)
- Based on the network layer
- Stateful/permanent
- Backscatter

Game of Resource Exhaustion

Pick one:

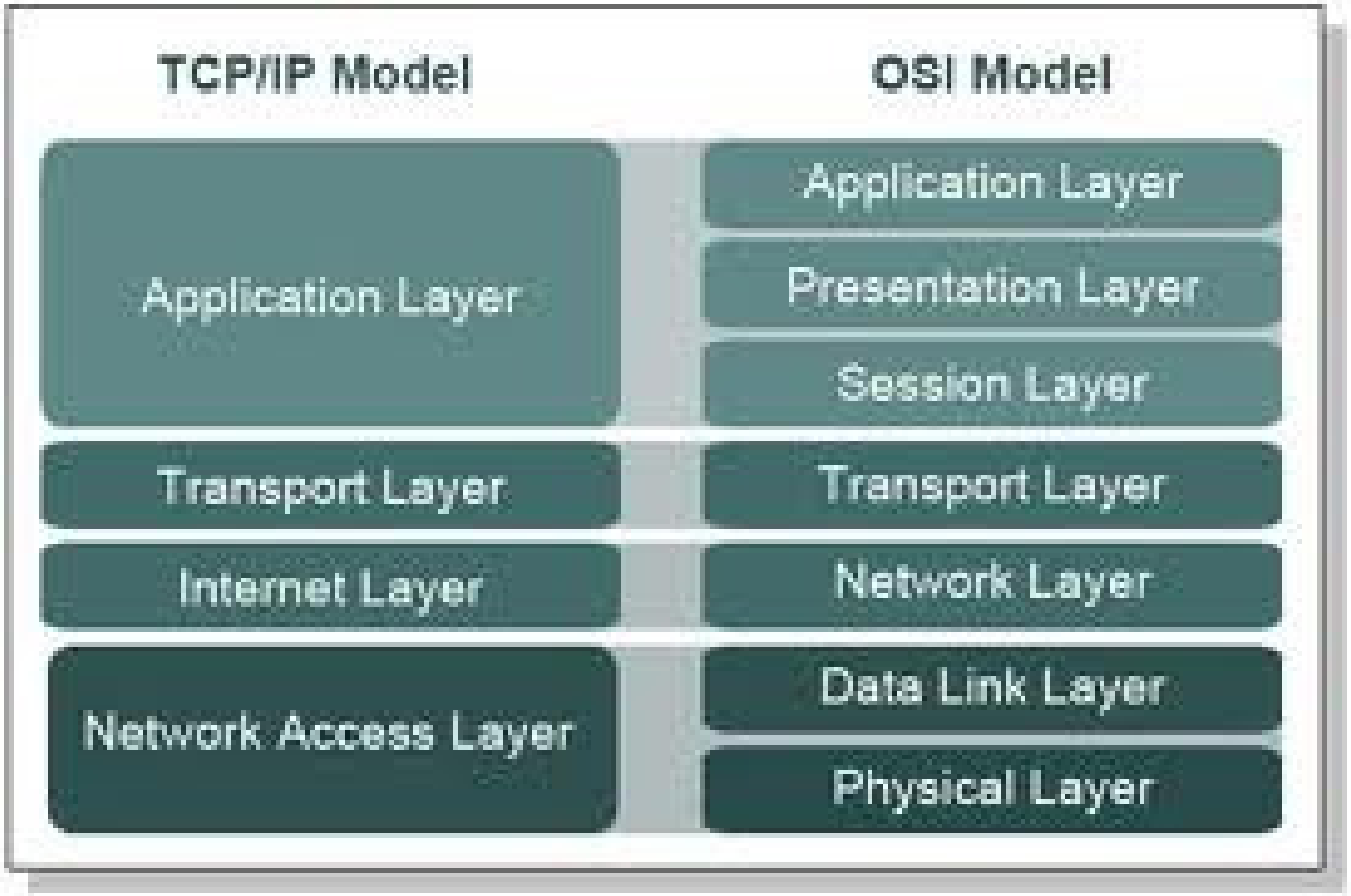
- Bandwidth
- PPS
- QPS
- Storage
- CPU
- Application specific (hardest) – could be any

…and only one is needed

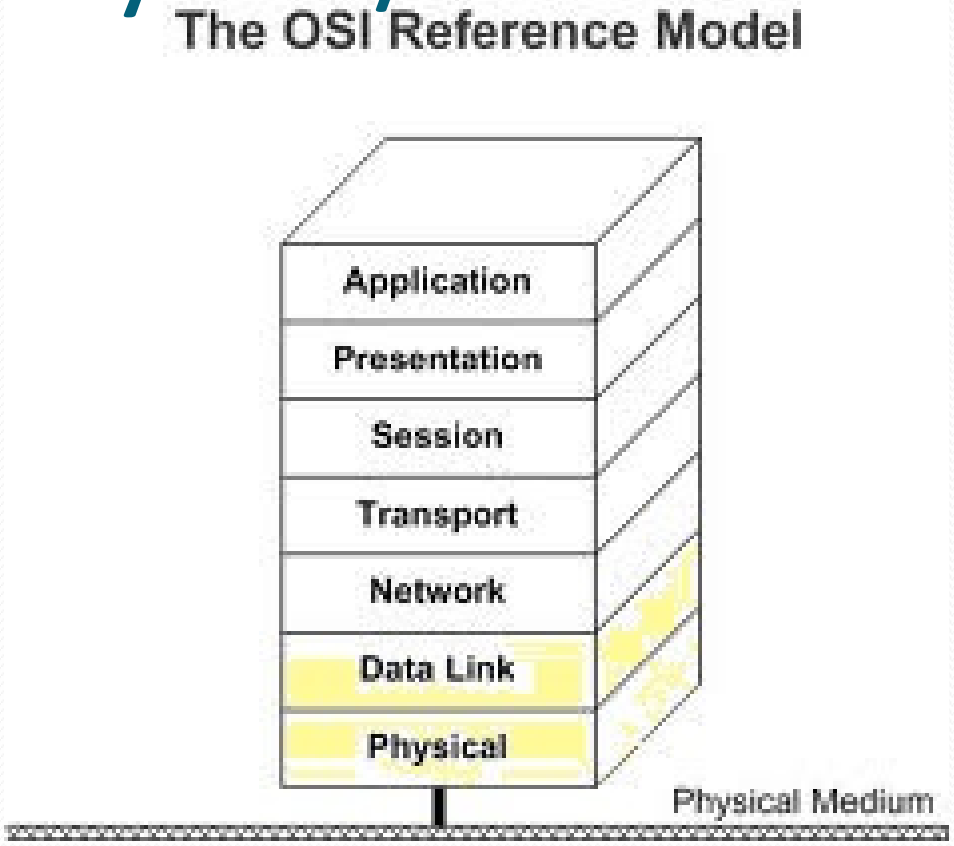
Another way to look is:

Last but not least – patience – Who gets tired first?

Attack surface (classification by layer)

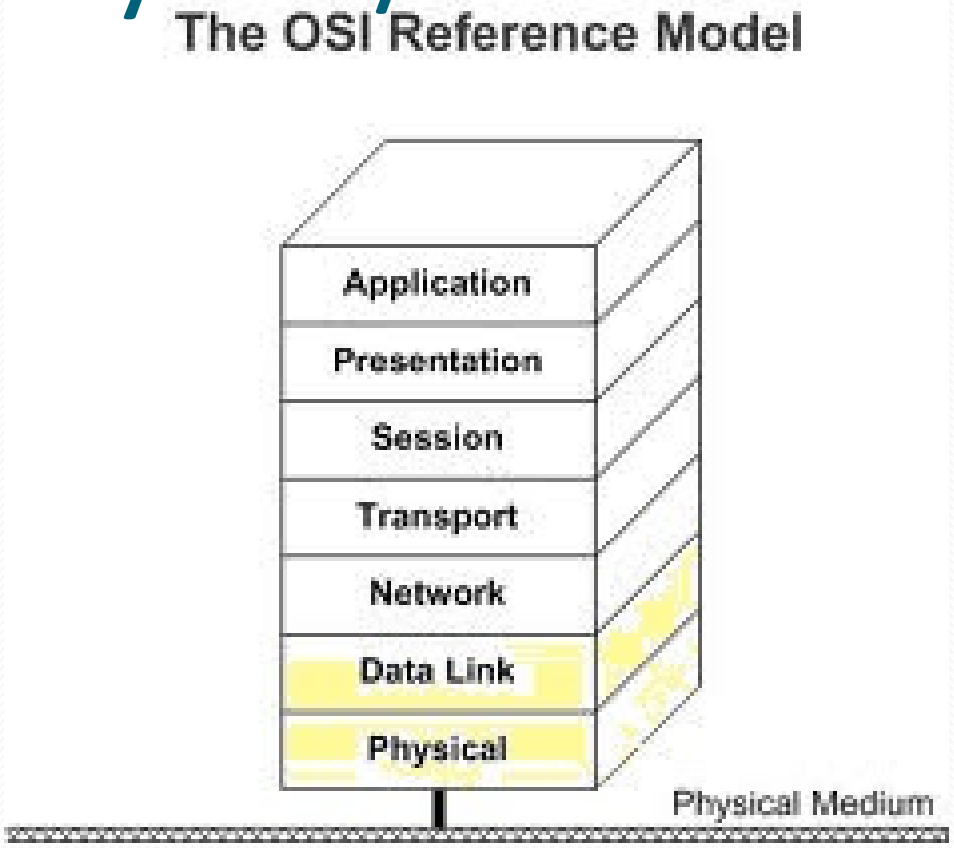


Layer 1/2 attacks



- Think?

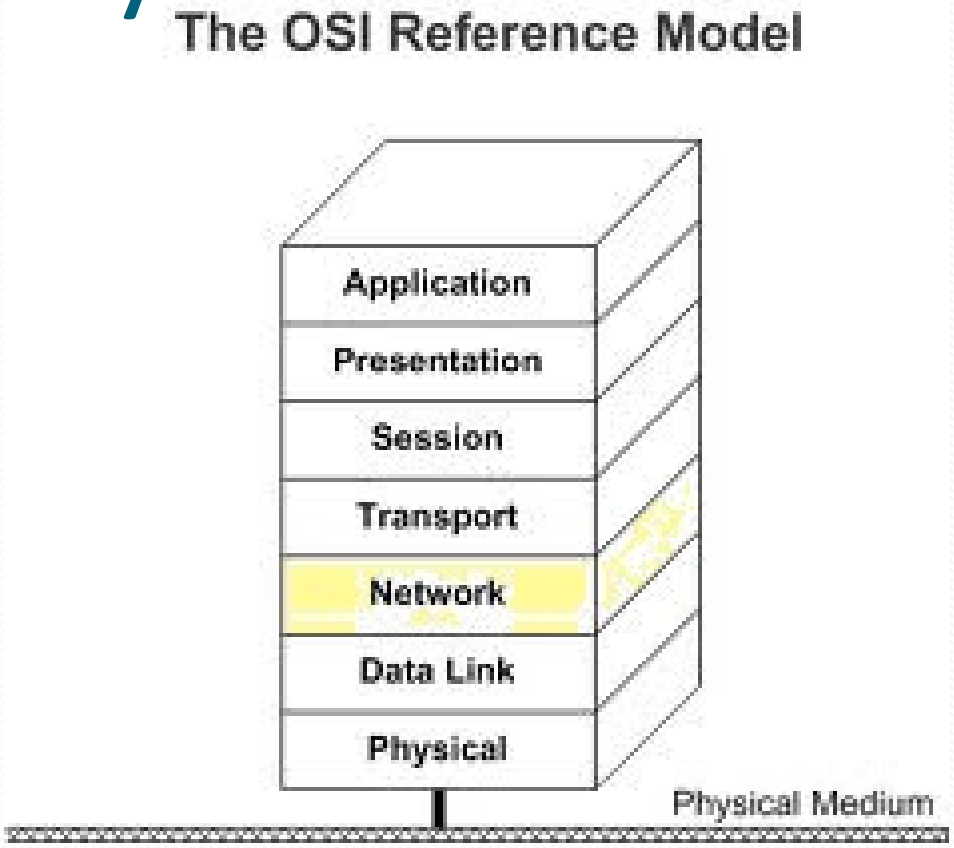
Layer 1/2 attacks



- Cut cables
- Jamming
- Power surge
- EMP

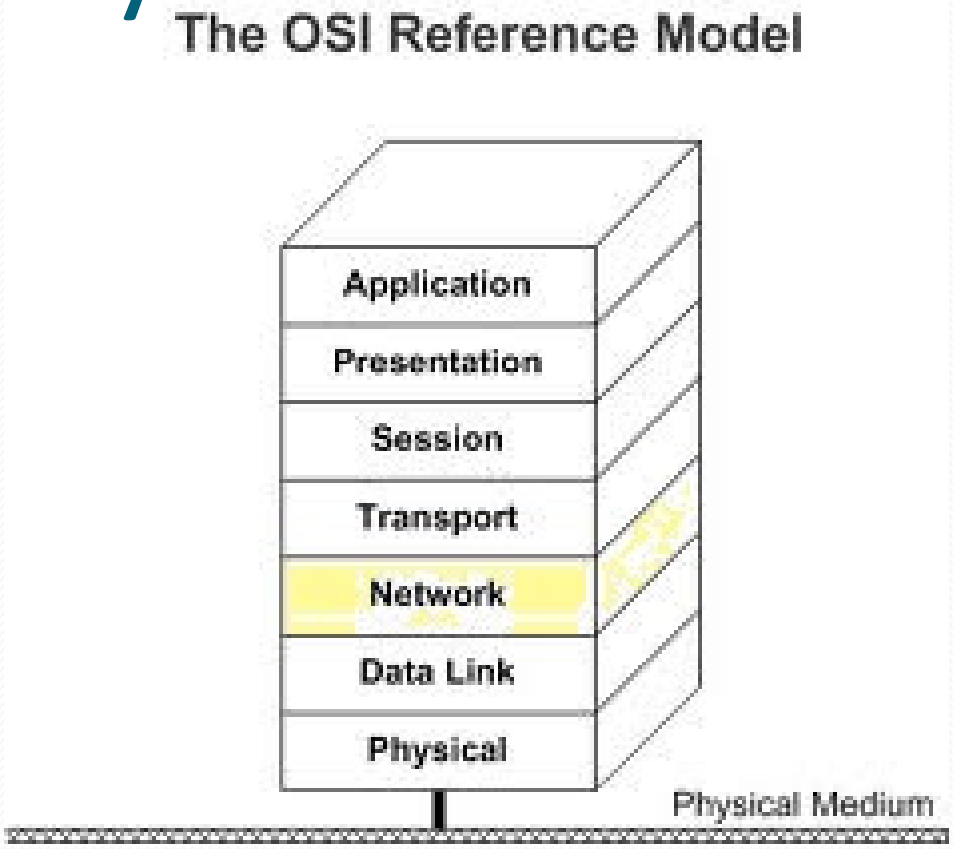
- MAC Spoofing
- MAC flood

Layer 3 attacks



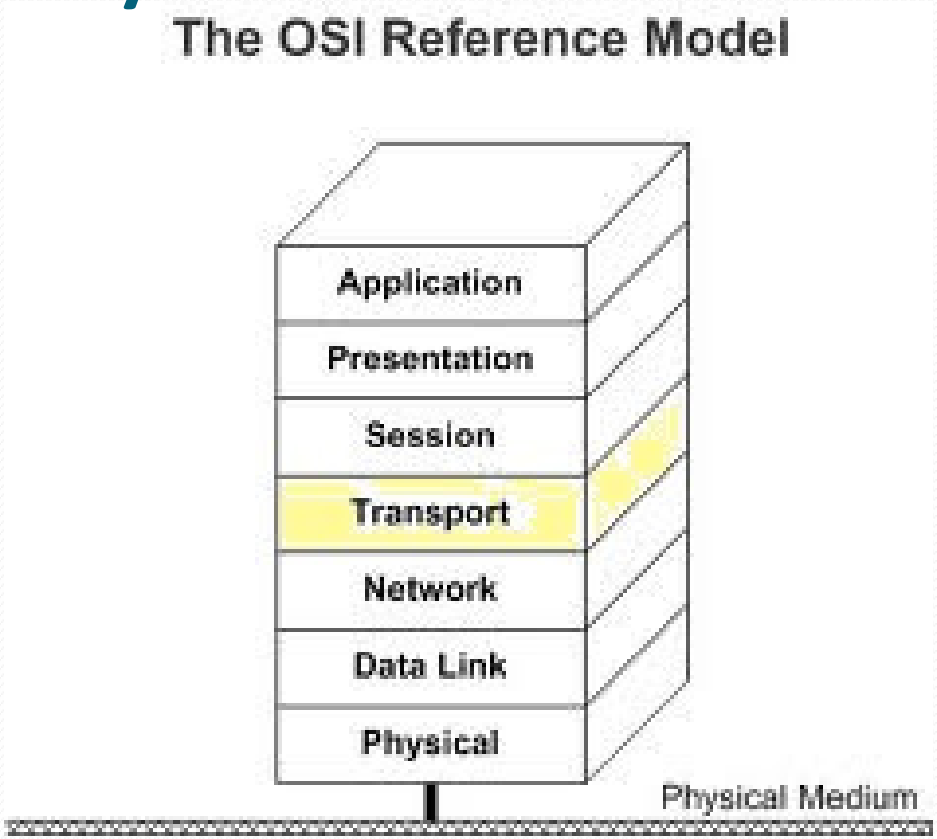
- Think?

Layer 3 attacks



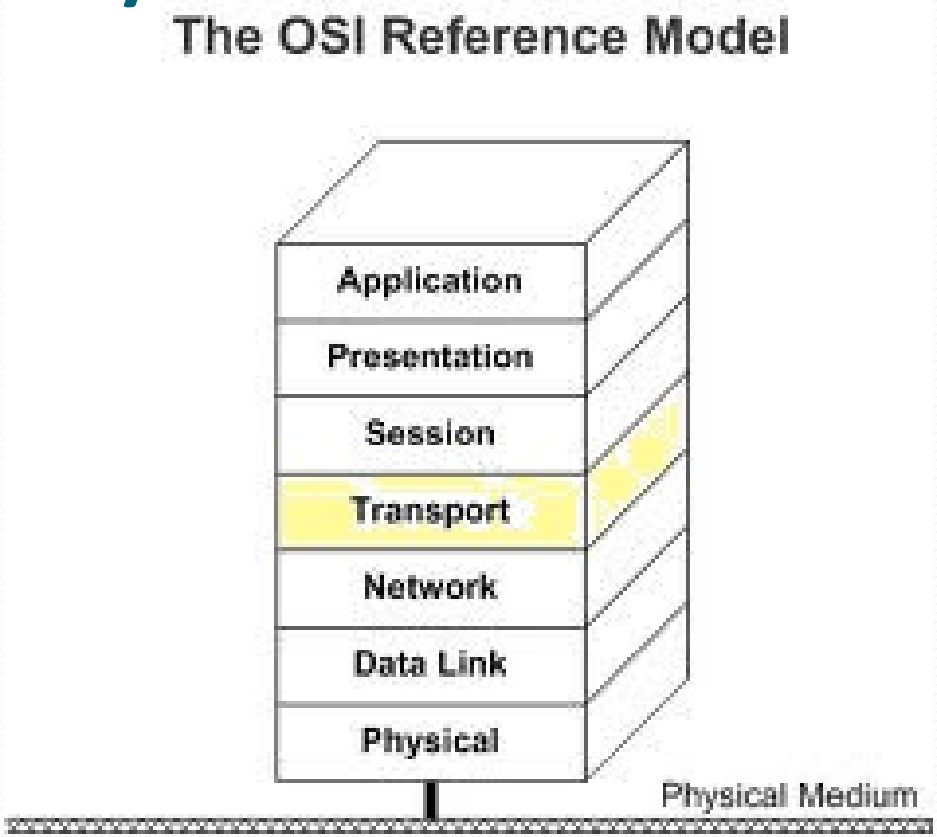
- Floods (ICMP)
- Teardrop
  (overlapping IP segments)

Layer 4 attacks



- Think?

Layer 4 attacks



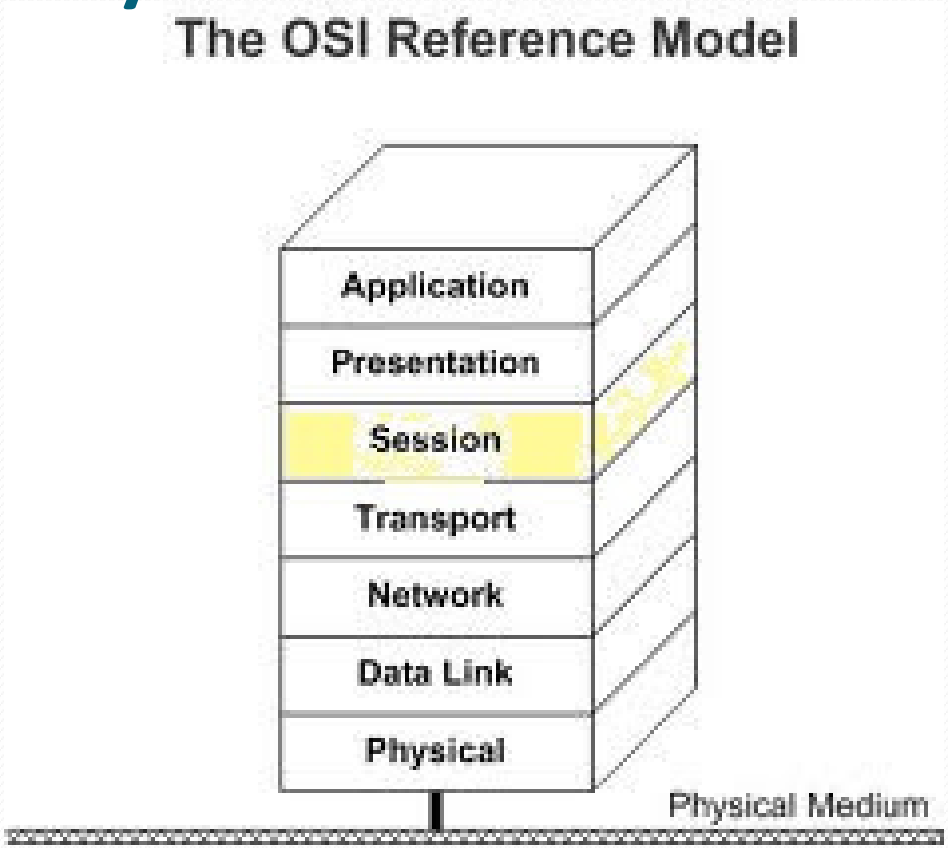
- SYN Flood
- RST Flood
- FIN Flood
- You name it…

- Window size 0
  (looks like Sloworis)

- Connect attack

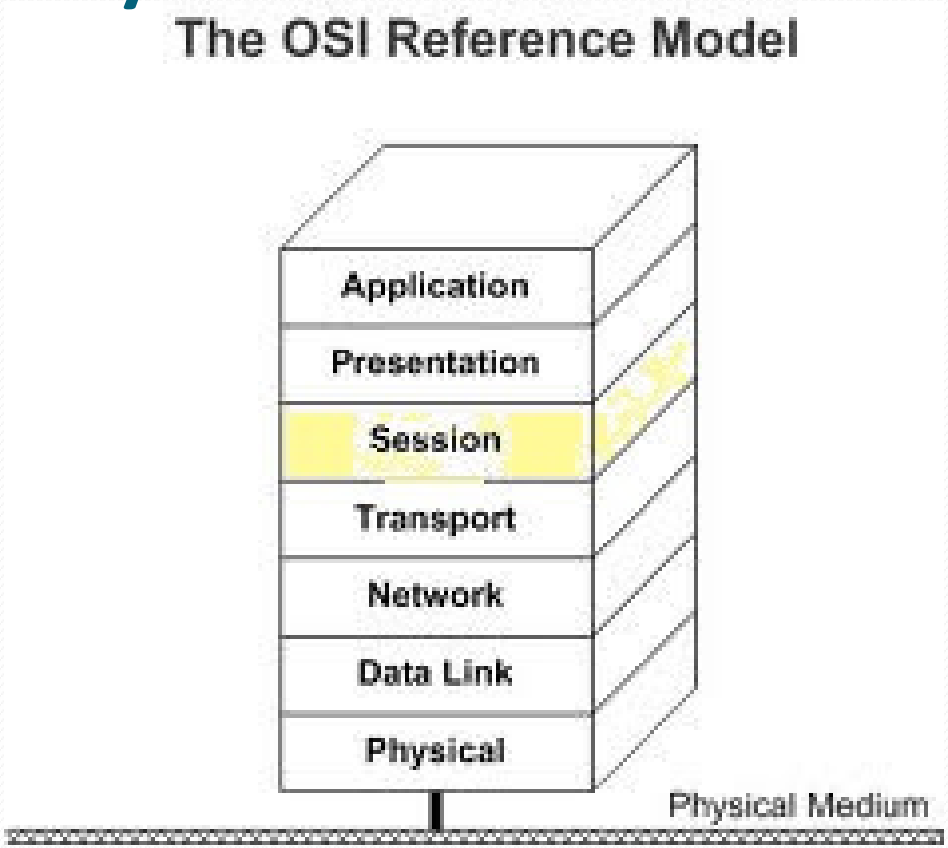
- LAND (same IP as src/dst)

Layer 5 attacks



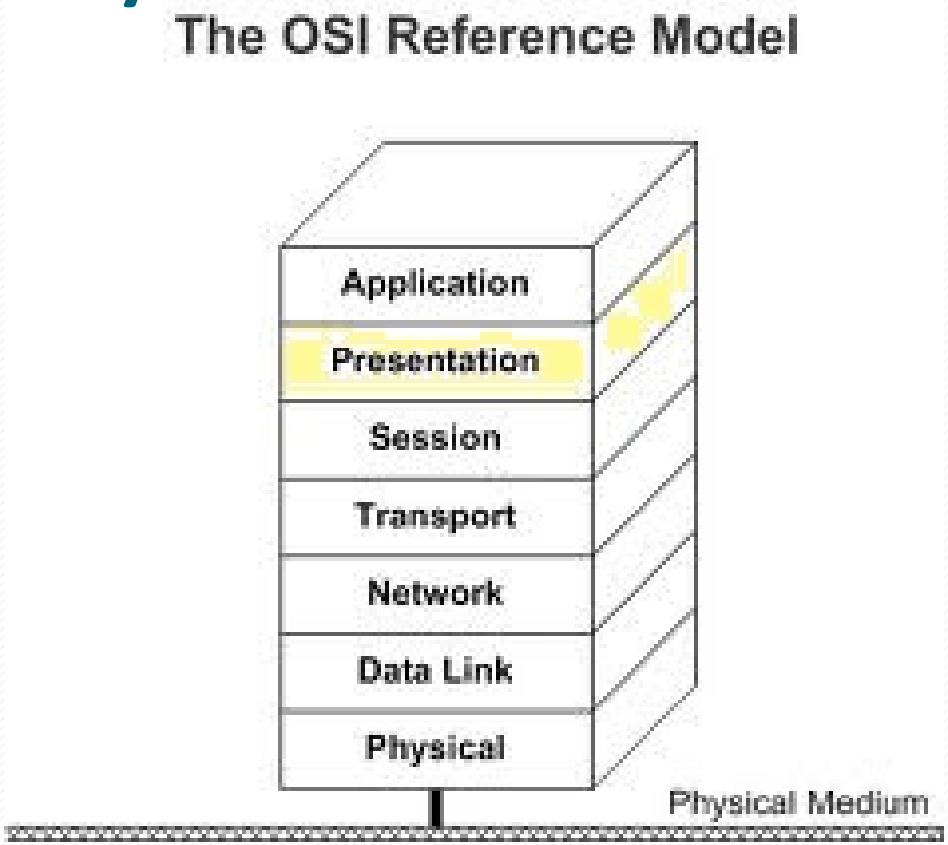
- Think?

Layer 5 attacks



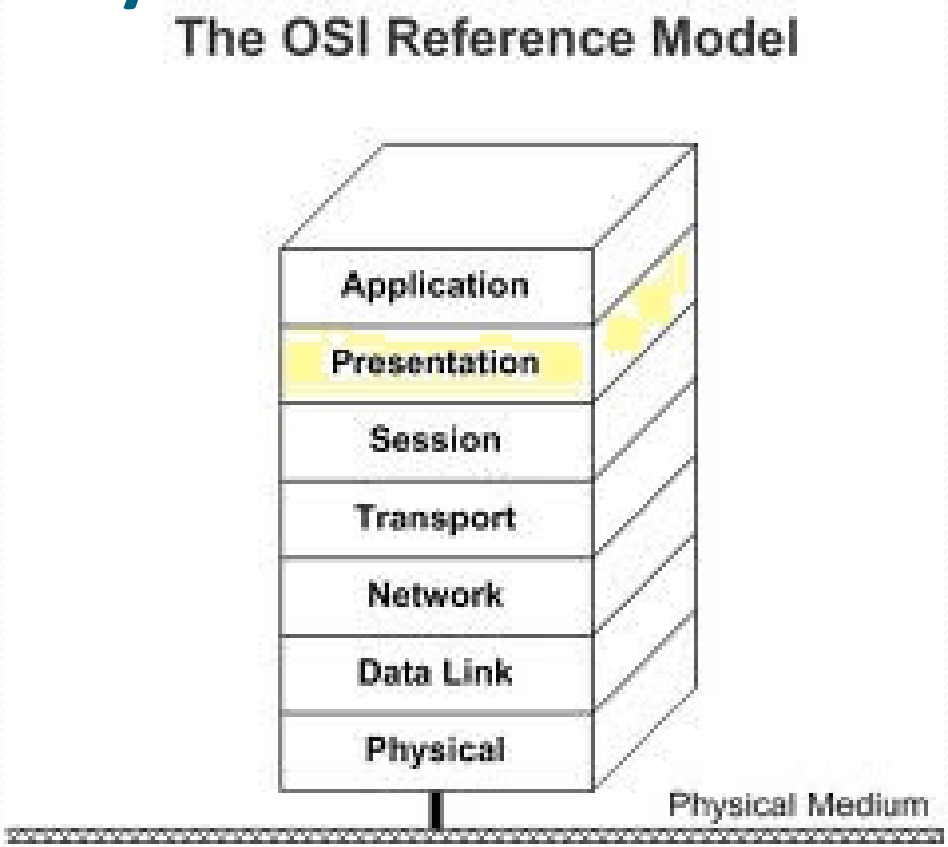
- Sloworis
- Just send data to a port with no NL in it
- Send data to the server with no CR

Layer 6 attacks



- Think?

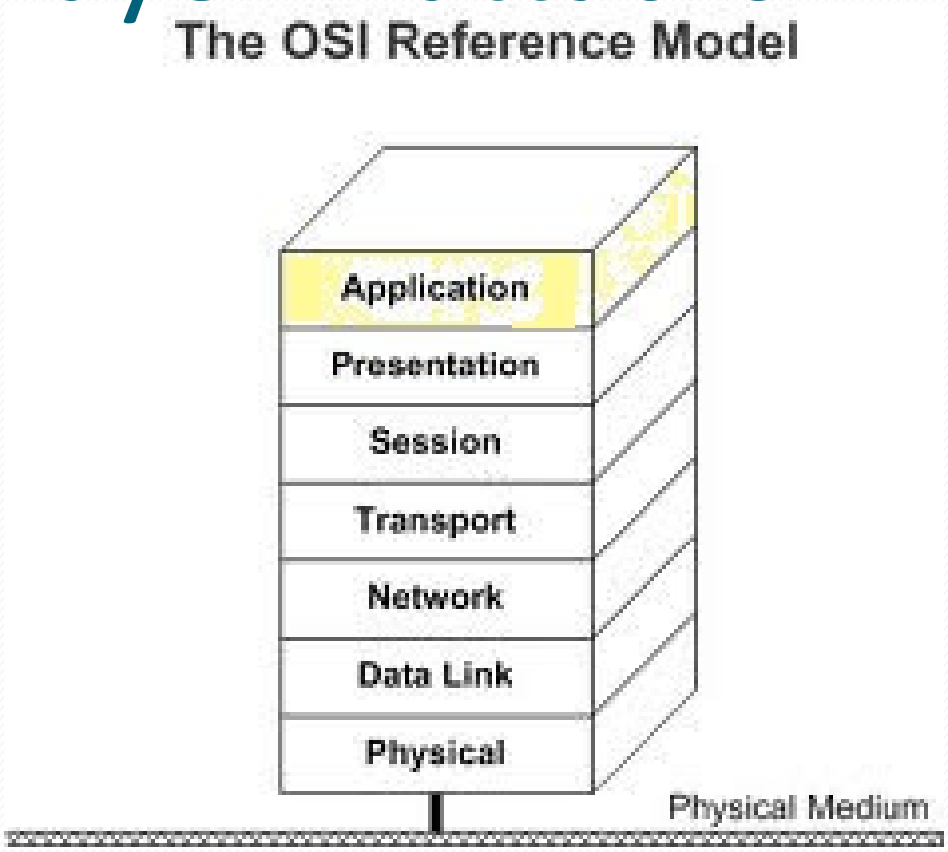
Layer 6 attacks



- Expensive queries
  (repeated many times)
- XML Attacks

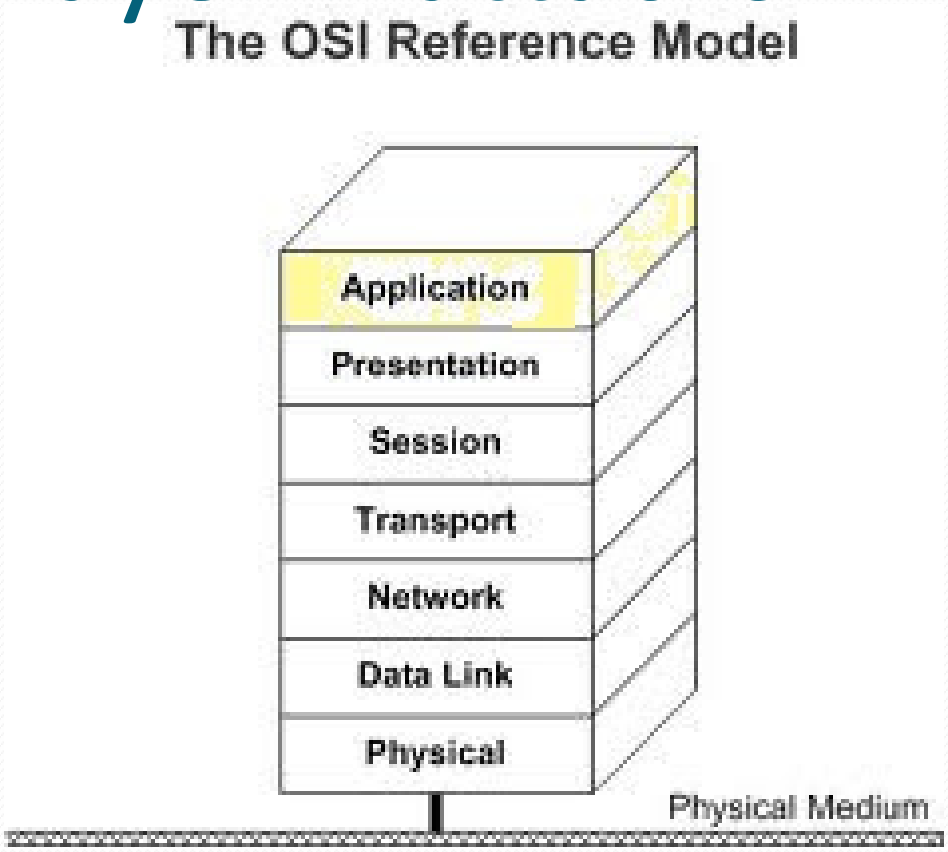
```
<!DOCTYPE lolz
[
<!ENTITY lol1 "&lol2;">
<!ENTITY lol2 "&lol1;">
 ]>
<lolz>&lol1;</lolz>
```

Layer 7 attacks



- Think?

Layer 7 attacks

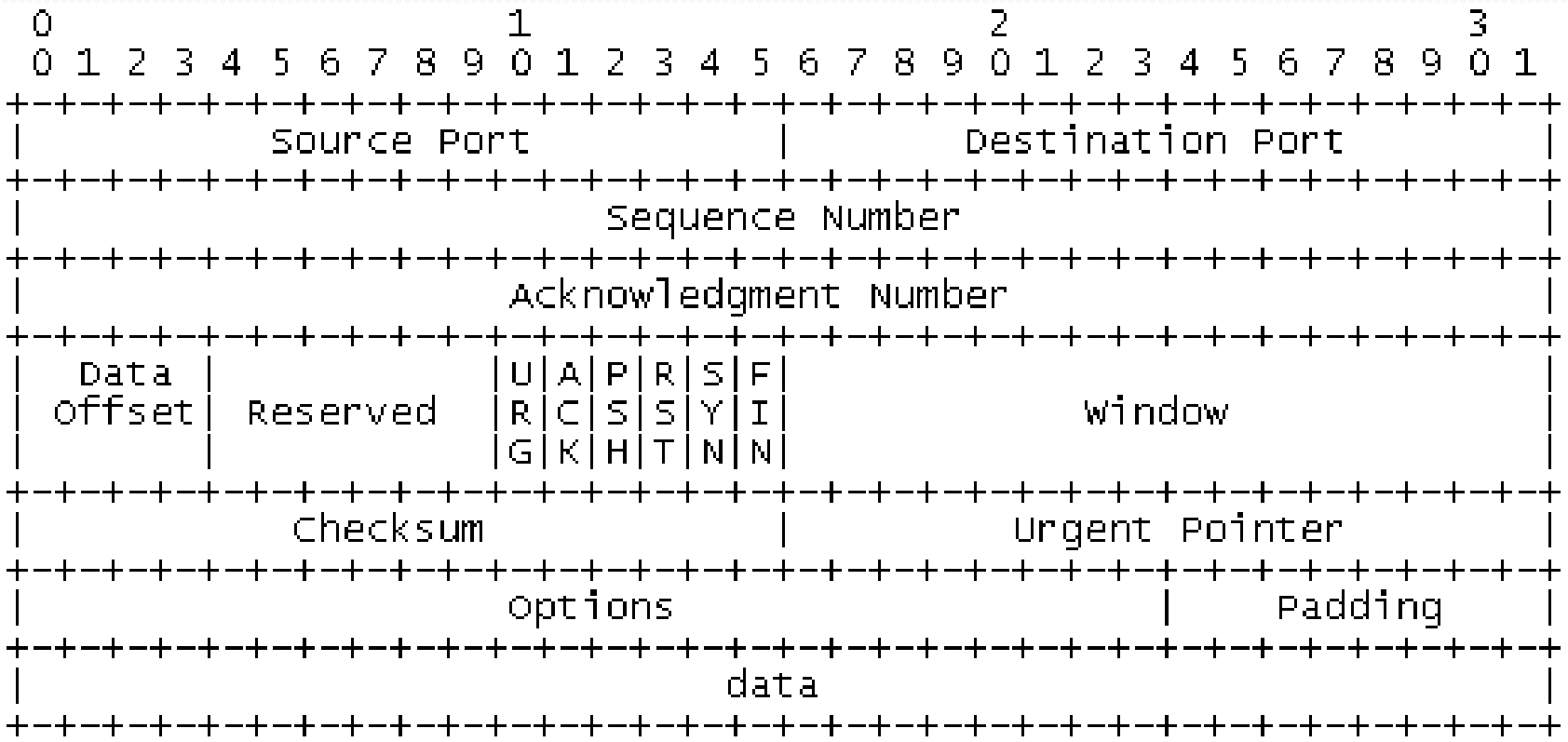


- SPAM?
- DNS queries
- Black fax

Intro to TCP

RFC: 793 / September 1981

TRANSMISSION CONTROL PROTOCOL



Simplified TCP state machine

- LISTEN – waiting for a connection request
- SYN_RECV – received request still negotiating
- ESTABLISHED – connection working OK
- FIN-WAIT1/2 – one side closed the connection
- TIME-WAIT – waiting for a while…
  - What is MSL?

Life of a socket

- Socket = TCP/UDP port + IP address

- Normal connection

```
[root@knight ghost]# netstat -nap | grep 12345
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      2903/nc

[root@knight ghost]# netstat -nap | grep 12345
tcp        0      0 127.0.0.1:12345         127.0.0.1:49188         ESTABLISHED 2903/nc

[root@knight ghost]# netstat -nap | grep 12345
tcp        0      0 127.0.0.1:49188         127.0.0.1:12345         TIME_WAIT   -

[root@knight ghost]# netstat -nap | grep 12345
[root@knight ghost]#
```

Detection on the host

- Your best friend: netstat

  netstat -nap

- Your next best friend: tcpdump

  tcpdump –n –i <*interface*> –s 0 –w <*target_file.pcap*> –c <*packet_count*>

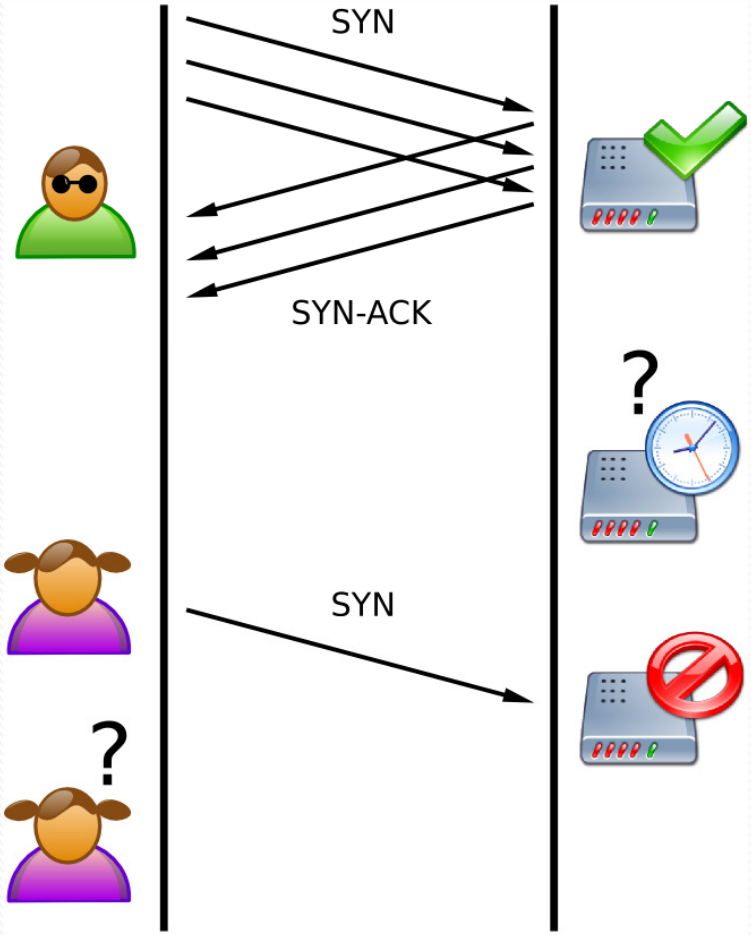
- Dedicated IDS (snort/suricata)

Mitigation

- Depends on the layer of attack
- Depends on the resource affected

- Do it yourself
  - dedicated hardware
  - Tune (change) your software
- Scrubbing providers
- Firewalls and challenges
- Horizontally scaled server frontends
  - “To the cloud!” ;)

SYN Flood



- What does it take:
  - Think 3-way handshake
  - Server has a number of slots for incoming connections
  - When slots are full no more connections are accepted

How to recognize SYN flood?

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	1339/rpcbind
tcp	0	0	0.0.0.0:33586	0.0.0.0:*	LISTEN	1395/rpc.statd
tcp	0	0	192.168.122.1:53	0.0.0.0:*	LISTEN	1962/dnsmasq
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN	1586/cupsd
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	2703/sendmail: acce
tcp	0	0	127.0.0.1:25	127.0.0.1:49718	SYN_RECV	-
tcp	0	0	127.0.0.1:25	127.0.0.1:49717	SYN_RECV	-
tcp	0	0	127.0.0.1:25	127.0.0.1:49722	SYN_RECV	-
tcp	0	0	127.0.0.1:25	127.0.0.1:49720	SYN_RECV	-
tcp	0	0	127.0.0.1:25	127.0.0.1:49719	SYN_RECV	-
tcp	0	0	127.0.0.1:25	127.0.0.1:49721	SYN_RECV	-
tcp	0	0	127.0.0.1:25	127.0.0.1:49716	SYN_RECV	-

…

SYN Mitigation

- SYN Cookies
  - Special hash
  - Enable by:
    echo 1 > /proc/sys/net/ipv4/tcp_syncookies
  - Other timeouts to tweak (in /proc/sys/net/ipv4/):
    tcp_max_syn_backlog

    tcp_synack_retries

    tcp_syn_retries

SYN mitigation (cont'd)

- SYN Proxy (TCP Intercept active)
  - Terminates at device/opens a second connection
- TCP Intercept passive/watch – sends reset
  - Resets the connection after a timeout
- Hybrid
  - Dynamic white lists

What is a SYN Cookie

- Hiding information in ISN (initial seq no)
- SYN Cookie:
  Timestamp % 32 + MSS + 24-bit hash
- Components of 24-bit hash:
  - server IP address
  - server port number
  - client IP address
  - client port
  - timestamp >> 6 (64 sec resolution)
- What's bad about them?

How to recognize socket exhaustion?

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	1339/rpcbind
tcp	0	0	0.0.0.0:33586	0.0.0.0:*	LISTEN	1395/rpc.statd
tcp	0	0	192.168.122.1:53	0.0.0.0:*	LISTEN	1962/dnsmasq
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN	1586/cupsd
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	2703/sendmail: acce
tcp	0	0	0.0.0.0:1241	0.0.0.0:*	LISTEN	1851/nessusd: waiti
tcp	0	0	127.0.0.1:25	127.0.0.1:60365	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60240	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60861	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60483	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60265	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60618	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60407	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60423	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60211	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60467	TIME_WAIT	-
tcp	0	0	127.0.0.1:25	127.0.0.1:60213	TIME_WAIT	-

Mitigation socket exhaustion/connect

- Enable socket reuse

  echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle

  echo 1 > /proc/sys/net/ipv4/tcp_tw_reuse

- Check learn about the value in

  /proc/sys/net/ipv4/tcp_*

- MSL decrease (on LBs) to a few seconds

Mitigation upper layers

- Architecture of applications
  - Apache – process based – In Linux kernel level threads
  - Nginx – event based
- Nginx
  (pronounced “engine x”) http://www.nginx.net/
- Mitigation through challenges
  Nginx plugin – Roboo (ECL-LABS.ORG)

How to DoS

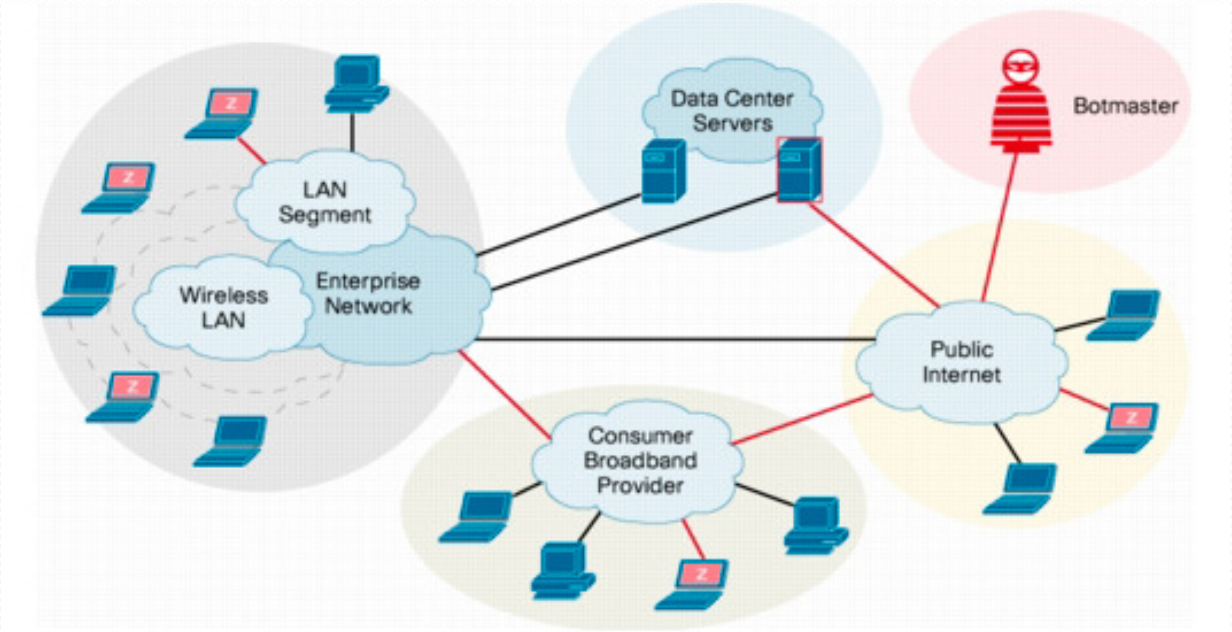
- Click really, really fast the "retry" button

- Scale hierarchically => recruit your friends/kids to do so

- Scale horizontally => get a botnet

Botnet components

- C&C (Command and Control)
- Proxy layer (optional) – think NginX ☺
- Bots/drones (any machine could be a drone)



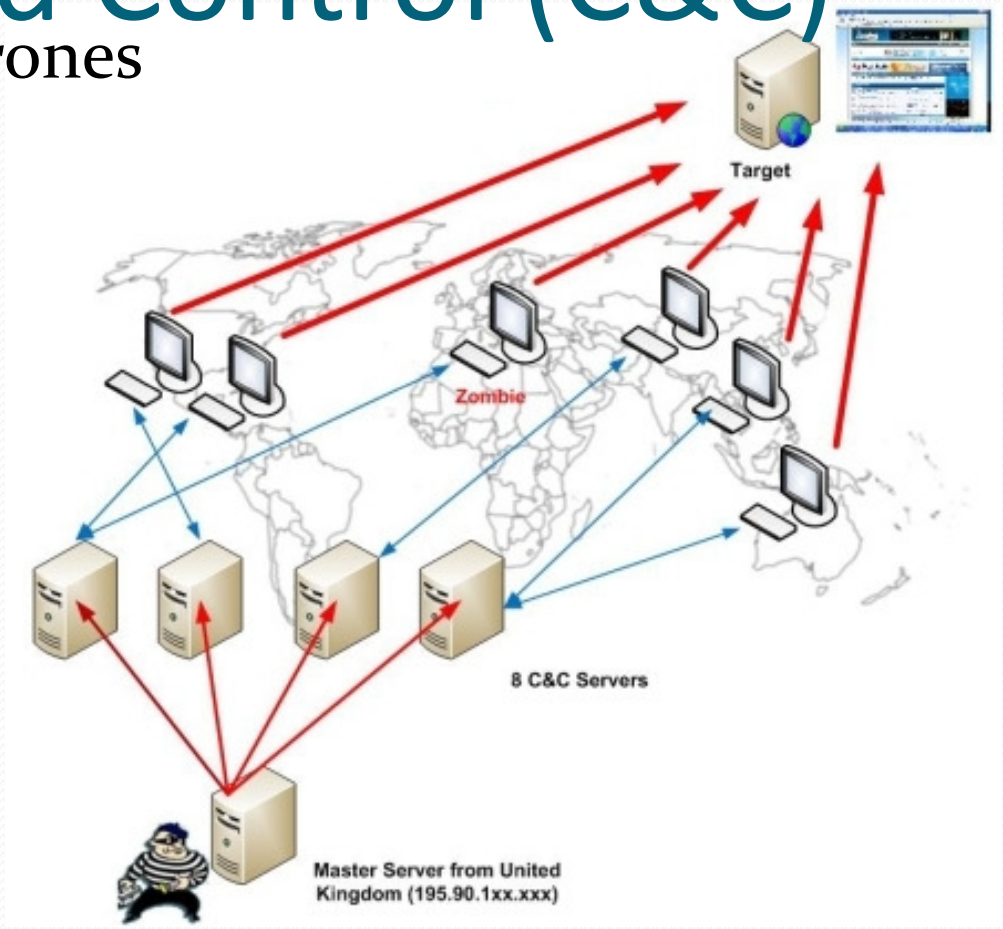
Drones

- Usually malware
  - Multiple ways of infection

- Rarely Opt-In (Anonymous)
  - User needs to download software
  - User needs to point it to target
  - Sometimes targeting can be automated

Command and Control (C&C)

- Attack is issued the drones read it and execute
- Scalability issues
- Inertia



You always have friends (find them!)

- Look around, who else might be suffering this?
- Build partnerships
- Build social contacts
- Prepare before it hits
- Be prepared
  so your ISP
  suffers before you

Tools to remember

- netstat
- tcpdump / wireshark

What can I do about it?

- RFC 2827/BCP 38 – Paul Ferguson
  - If possible filter all outgoing traffic and use proxy
- Patch your systems
- Learn how to use
  - tcpdump/wireshark
  - netstat
- Check out the Arbor Networks Report http://www.arbornetworks.com/report

Q&A

December 9-11

Mountain View, CA

http://www.baythreat.org/