

# Network Security 1: Introduction

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Timetable Weirdness

**Tues 1400 G33, Wed 0900 (sorry!) Arts 125. But...**

<b>19th January</b>	Extra Lecture, 0900 Arts 223
<b>23rd January</b>	No Lecture
<b>24th January</b>	No Lecture
<b>29th January</b>	Extra Lecture, 0900 Arts 223
<b>6th March</b>	Lecture is in Aston Webb WG5
<b>20th March</b>	Lecture is in Aston Webb WG5

# What this course is

- Enough network security that you could make a reasonable job of securing an enterprise network...
- ...except you might need someone to help with the technical details of the precise hardware you are using.
- Emphasis on the **what** and the **why**, rather than the **how**. This is not a CCIE class.
- But we will get practical in some exercises.
- Caveat: 10 credits, right?

# Requirements

- We are going to get our hands dirty with some Unix (Linux, although you can use MacOS or Solaris if you want, because pf and ipf are cool)
- It would be very, very helpful if you have a laptop which can run medium-size virtual machines, ideally two at a time.
- No programming requirements, although being able to script in bash/perl/python would be useful.
- I am assuming you've done my Networking course or similar, to the point you're familiar with TCP at a packet level.
- I'm also assuming you've taken either Secure System Management or Tom's second-year course, but I'll attempt to fill in the gaps for those of you that haven't. Hands up...?

# What you will learn

- How to assess what needs securing (to a point)
- How to secure it
- How to test and evaluate the security
- Some realism in what does and does not work

# Week 1

- All of this is slightly tentative as I want to make space for a much more in-depth look at webserver security than I have done in previous years.
- This introduction
- Assets, Threats, Risks: recap and focus on networks and network connected equipment

# Week 2

- Host security: logging, patching, service minimisation
- Host security: least privilege, isolation, jails, zones, virtualisation, containers, etc

# Week 3

- Defence in depth vs. multiple shallow defences. Attack trees. Side-channel and subsidiary protocols (some of you have seen some of this content before in SSM).
- Make a start on firewalls to introduce the exercise.
- **NB: actually Friday of Week 2, and Monday of Week 4.**

# Week 4

- Firewalling: host, network. Design issues. Testing. IPv6 issues. Statefullness.
- Firewall implementation. Linux v everything else (IP stack v interfaces)

# Week 5

- Network IPS/IDS, concepts and limitations. Host IDSSes, Tripwire, etc.
- Admission control, Wireless authentication, Radius for 802.1x, WPA2/PSK, WPA2 Enterprise.

# Week 6

- Data diodes, proxies, other firewall alternatives.
  - Some of this is necessarily speculative as data diodes are mostly custom for specialist customers.
- Web Server Security (HSTS, Content-Security-Policy).

# Week 7

- Application security: application design, firewall friendliness, NAT friendliness
- Application security: TLS and authentication via certificates. OTP tokens. Oauth (maybe).

# Week 8

- Denial of service, amplification attacks, mitigations, egress filtering for “good network citizens”
- Attacks on DNS
- Certification issues (HPKP and its problems, CAA, etc).

# Week 9

- VPNs: concepts and components
- Probably start on IPsec

# Week 10

- IPsec: concepts and components
- IPsec: Key exchange

# Week 11

- VPNs: protocols (OpenVPN/SSL-VPN, IPsec, extensions to IPsec, IPv6 issues)
- Recap, exercise tutorial, spare

# Books

- More resources on the net than in print
- Still recommend the Nutshell book “Practical Unix and Internet Security” by Garfinkel and Spafford, even though the technology is largely out of date.
- Also “Stalking the Wily Hacker” by Cheswick and Bellovin, with the same caveat.
- Ross Anderson’s “Security Engineering” (2nd Edition) is always worth reading

# Assessment

- Due 12th February: building a firewall and testing it
- Due 12th March: IPsec configuration (probably, but possibly Content-Security-Policy depending on practicality)
- In each case, two week deadline, and a lecture of feedback and discussion.

# Office Hours

- 1000–1200 Tuesdays and Wednesdays, CS room 132
  - Or mail me and make an appointment
  - Or just bang on my door (I now have a less scary photograph, apparently)

# Lab Session

- 1100–1200 Mondays, UG04, when there is an exercise live and/or something useful to do.
- This clashes with Advanced Crypto: hands up? Proposals?

# Timetable Weirdness

**Tues 1400 G33, Wed 0900 (sorry!) Arts 125. But...**

<b>19th January</b>	Extra Lecture, 0900 Arts 223
<b>23rd January</b>	No Lecture
<b>24th January</b>	No Lecture
<b>29th January</b>	Extra Lecture, 0900 Arts 223
<b>6th March</b>	Lecture is in Aston Webb WG5
<b>20th March</b>	Lecture is in Aston Webb WG5

# Network Security 1: Threats, Assets, Risks

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Why Network Security?

- Special case of security in general, and in principle 27001 (or similar) picks it up the same as it picks up door locks and disk disposal.
- But has a much longer history: I published in a technical journal about firewalls in (I think) 1993, when security as a specialism was restricted to banks and was mostly about locks.
- We were worrying about securing networks before we were worrying about security more general, because of the Morris Worm (November 1988).

# It's the Internet

- Network security's history is mostly about the area around, and fanning out from, your internet connection(s).
  - There were security issues around inbound dialup modems, but they were largely ignored.
  - JANET security, pre-Internet, was very weak.
  - A lot of the discussion was, and often still is, about the design of the area around the router that the connection arrives on: firewalls and DMZs.
  - In 2018, we need to look deeper and wider.

# What are we protecting?

- Confidentiality
  - We don't want unauthorised people reading our data
- Integrity
  - Or changing/deleting our date
- Availability
  - And we want the computers to keep working

# What are the assets?

- Information assets are both data and services.
- So that's files in filestores, email in email repositories, private webpages in intranet servers...
- And also the fileservice, the email service, the web service, the routers that glue it all together, the DNS servers, the DHCP servers...

# Who are the threat actors?

- When considering security, and information security, in the large, they are almost all **insiders**. We have the advantage of physical security.
- But for network security, the threat actors will be in large part **outsiders**. They may have help from inside, but internal attackers are a smaller part of the problem.
- This course focuses on dealing with outsider risk, but good practice helps in all scenarios.

# Types of actors

- “Script Kiddies”
- “Anonymous” and their equivalent
- Well-funded state actors
  - We need to consider motivation, skill and resources.

# Types of threat

- Service disruption
- Information theft
- Service theft
- Fraudulent changes to data
- Dot, dot, dot.

# Types of risk

- DDoS
- Theft of credentials
- Malware and Phishing attacks; Viruses/Trojans
- Bribery, blackmail
- Cryptographic attacks
- Service compromise
- Again, dot, dot, dot.

# What is our objective?

- To secure data so that it can be **trusted** by authorised users.
  - Data that is not confidential doesn't need to be confidential, but probably does need to be unchanged and available.
  - In general, integrity and availability are often more important.

# Identifying Assets

- Ideally, we start from the 27001 asset register that has been constructed as part of our fully accredited ISMS.
- But more often, we have to be pragmatic: “we haven’t done 27001 so can do nothing” is not a good position to try.
- Naively, we are concerned with assets that are:
  - networked
  - potentially exposed to the outside world
- But we also need internal controls, both to limit insider threats and to prevent escalation of one vulnerability into a wider attack.

# Identifying Assets

- Typical sites may have a mail server, a web server, some sort of file sharing (Sharepoint, etc).
- But larger sites may have a large range of applications including databases of various flavours, test equipment, production lines equipment, ovens, **centrifuges**...
- The first question is “why am I exposing this to the network?”

# Threats and Risks

- Who might want access to an asset?
  - Topic for another course, but often confidentiality is much less of a problem than integrity or availability.
- What would they do with it?
- What would it cost you if they did it?
- What are the legal implications?
- How might they attack it?

# Assets aren't just information

- Routers and switches: if you can break into them, you might be able reconfigure the network to bypass security (a network design which survives attackers owning the switches is hard)
- Identification servers (active directory, etc): if you can break into them, you can add new users and give them privileges, or upgrade existing users.
- DNS Servers: a future lecture for the full horror story, but you can do Really Bad Things if you can reconfigure them or otherwise attack them.
- And see also centrifuges.

# **Slack for School of Computer Science (Thanks to Jonathan Duffy).**

Please join here or look on the front page of the Network Security Canvas module.

<https://goo.gl/Tz4v5d>

There is a channel for #network\_security

Great for collaborating, discussing problems and ideas etc

Feel free to create and join as many channels as you like

Used in many work places

Please join and install the apps

The more people using it the more useful it becomes



# The two extremes of protecting a network of computer

- Default Deny: pass all connections through a restrictive set of filters and proxies, essentially isolating the internal systems, while providing a small set of machines which communicate with the outside world.
- Default Permit: all internal systems have access to the internet, and are responsible for looking after their own security.

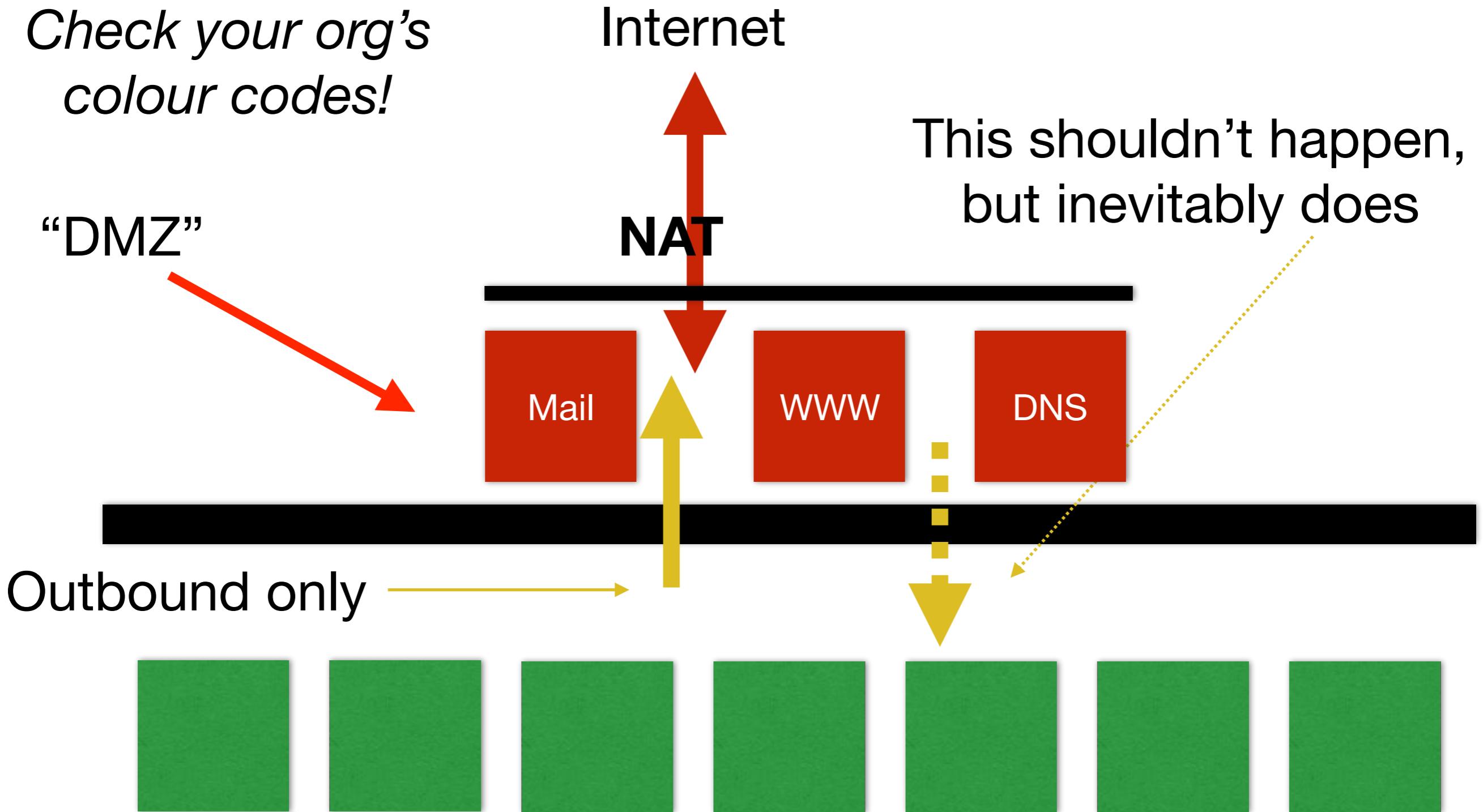
# Approach 1: Default Deny

- One approach is to completely separate everything from the Internet, and then expose a very limited subset of things that need to be available.
- Extremely difficult to do well, and operationally very painful.
- Data crosses the gap in all sorts of ways anyway.
- Companies try to do this and usually give up; much more common in government. Open question as to how well it works. Design pattern from the Clinton presidency.

# Default Deny

*Check your org's  
colour codes!*

“DMZ”



# Approach 1: Default Deny

- Idea is that outside world and inside world can contact machines sat in the DMZ, and the machines in the DMZ are hardened enough to guard temporary data and not be repurposed.
- Crucially, the internal firewall is one-way: it only permits connections initiated from the inside.
  - This is often forgotten, and also very hard to do in a real operational network. For example, both putting an IMAP server in the DMZ and allowing SMTP through are risky temptations.

# Approach 1: Default Deny

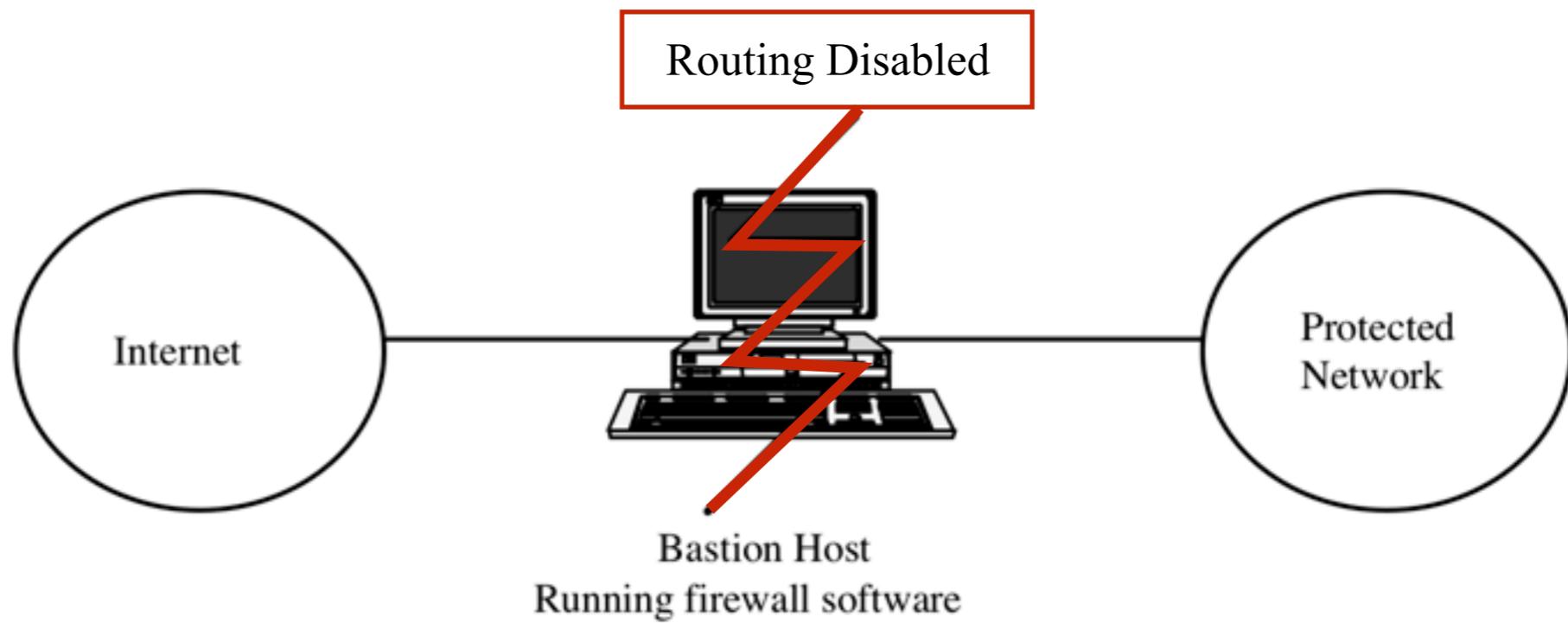
- Formalised by first internet touch point for White House circa 1992, done by Marcus Ranum of DEC (?), then “Trusted Information Systems”, presumably with support from NSA/spooks.
- The “TIS Firewall Toolkit” is still worth looking at, 25 years later. <http://www.fwtk.org/fwtk/docs/overview.pdf> (following diagrams come from that paper).
- Router-level policies **much** less capable then (no connection tracking, **no NAT**)
- Controlled outbound connections (in that era, ftp, mail and telnet) as well as inbound.
- Also, POP3 still main means for fetching email from servers, because no mobility (primary use-case for email fixed desktop device, GSM several years in the future, laptops several years in the future, Blackberry etc 10+ years in future).
- The world has changed a lot. The design pattern hasn’t, or at least hasn’t enough.

# Note interesting change in terminology

- Today, “firewall” has come to mean a box which sits between two networks, filtering packets (and by implication connections) which pass between them.
- When design patterns were first arising, “firewall” meant the whole area of one or more filtering (also “screening”) routers — with less flexible rules than today — and machines operating as proxies/bastions/etc, or it meant the proxies.
- Some of the slippery weakening of the design comes from this change in terminology: when people installed firewalls in the past they were much more complex and capable because of the computers involved. Now they’re just packet filters. Fancy packet filters, as we’ll see. But they only understand protocols insofar as they need to for NAT, and that is completely defeated by encryption.

# Note historic use of “firewall”

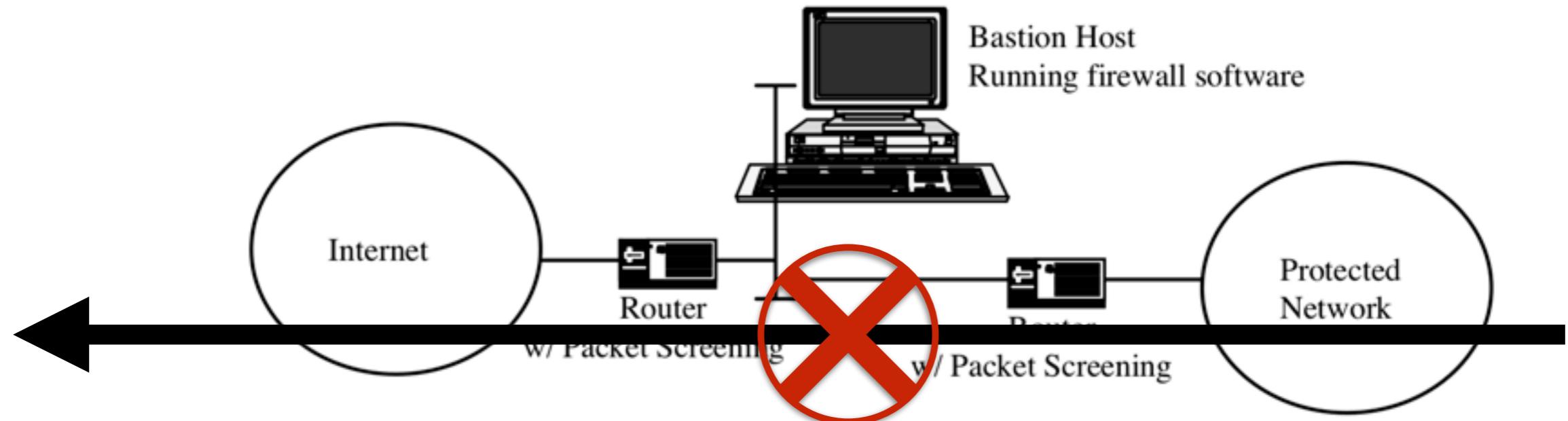
Figure 1: A dual-homed gateway



- Firewall element is the “bastion host”.
- No routing, no NAT. Log in to bastion, connect out from there, and (perhaps) vice versa.

# Modern setups were envisaged, but different

Figure 3: A Screened Subnet Gateway



- Again, firewall refers to the whole thing, or to the proxy software.
- Absence of NAT means no path In->Out for packets, everything is via the bastion host.

# UKUUG Winter Conference, 1996: many of you not born

- “The second day opened with “**Network security without firewalls**” by **Ian Batten**. This presentation documented Ian's experiences and opinions of firewalls, and why he believes they give no advantage for the system set-up he uses. He overviewed the advantages (firewalls do work, and can be very beneficial if administered properly) and disadvantages (complex to administer, can breed complacency) of firewall systems, and the alternatives to them. The disadvantage of the complacency issue was stressed. **There is a general attitude of “we've got a firewall, so we're OK as regards security”. This is terribly naive. For most systems, the greatest threat is from within. The percentage of adept penetrations from outside the system is very low in comparison with the percentage of data loss from users taking software and information off the system. The alternative system proposed was that of a screening router coupled with host security. So, all packets that are anomalous (eg claiming to be from that machine) or unnecessary (eg ICMP redirects) are denied, and internal measures are taken to help counter any external attack.** These include using encrypted ident as an auditing tool, using ssh for remote access to the system, and extensive system logging among others. The latter was stressed extensively, though care must be taken to ensure log files are resistant to tampering.”

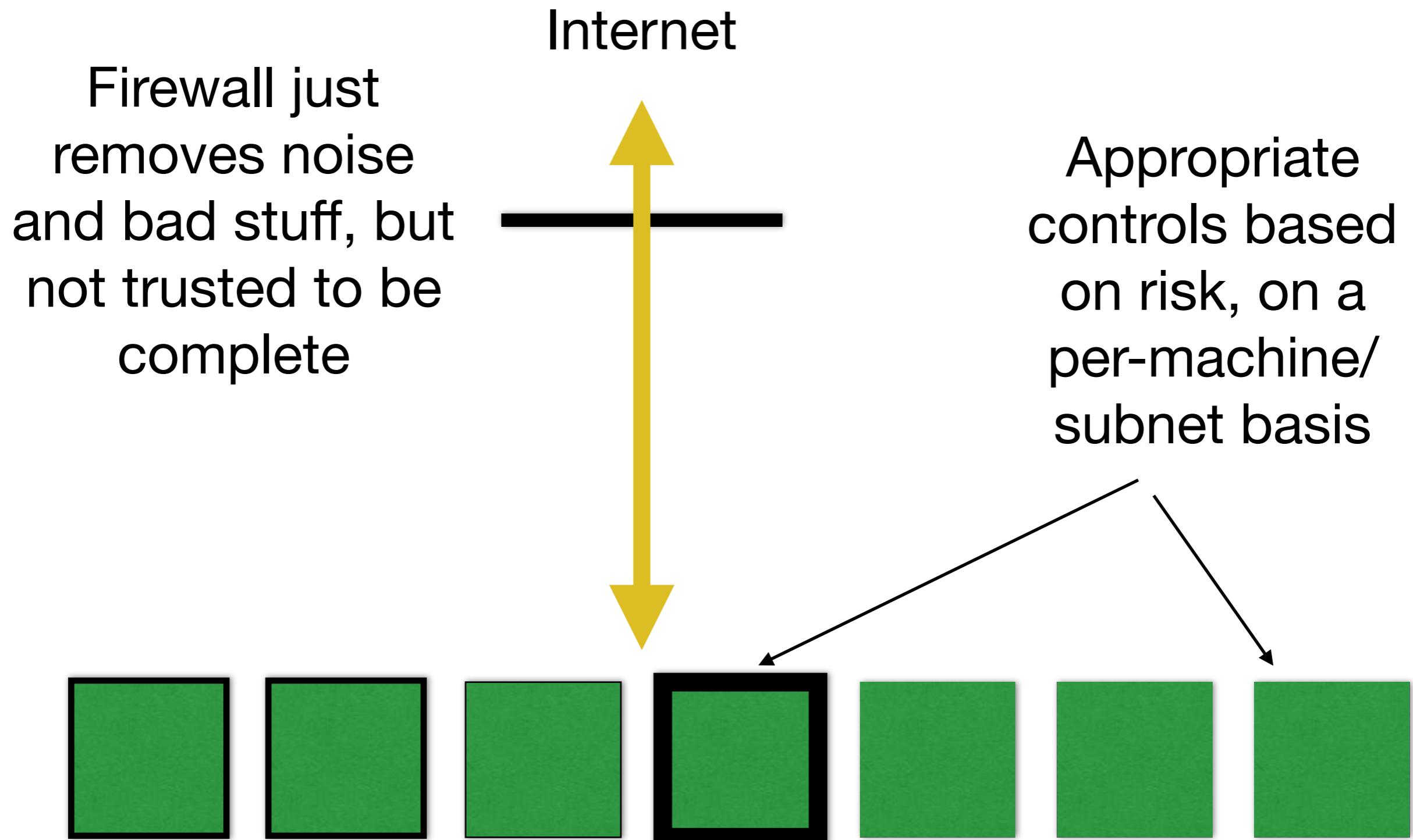
# 22 years later

- I was wrong about identd, and history has proven me wrong.
  - Idea was “connect to port on machine, provide a remote address and a local port, be told owner of that socket”. Relies on assumptions that were with hindsight broken even in 1996 (and probably in about 1993 when I started using it).
- I stand by the rest of the presentation. I have rescued a copy of the accompanying article from backups, so you can see the debates in this lecture are not new:
- <https://www.batten.eu.org/~igb/security.pdf>
  - also <https://goo.gl/bY3347>
  - Written in LaTeX 2.09, which handily modern xelatex processed without demur.
  - You might find it worth reading to get an overview of some of the issues, even at this distance in time.

# Approach 2: Default Permit

- The whole network is exposed to the outside world, with individual assets protected on a case-by-case basis. Network-wide protection relies on security policy, ideally enforced by Group Policy etc, often not enforced or enforced only by (or on!) paper.
- University networks would be an obvious example, but any business that is moving towards bring your own device is implicitly doing this.

# Default Permit



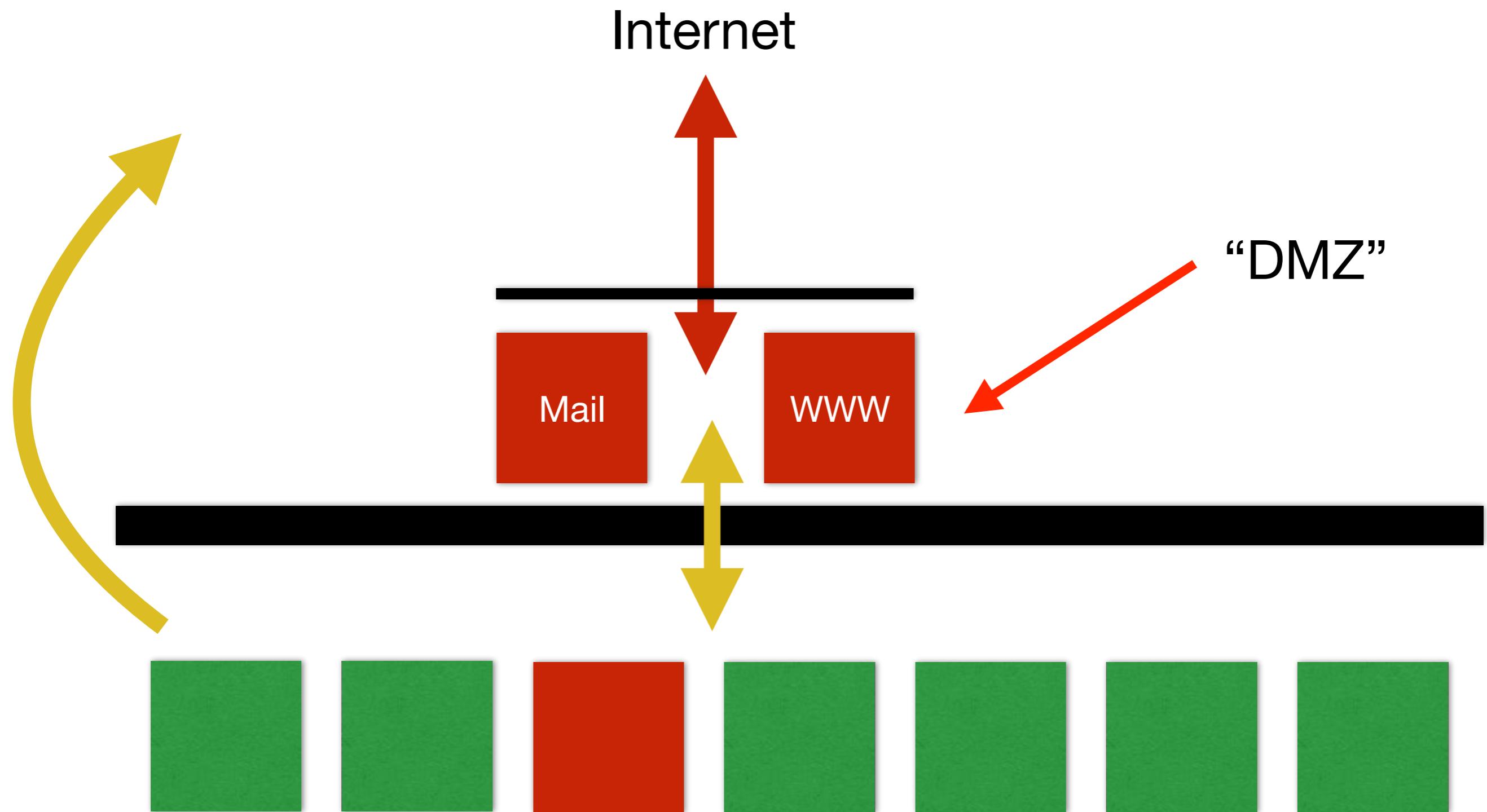
# Benefits of Default Deny

- You don't need to inventory all your assets, just the stuff that is exposed to the outside world.
- You can rely on the inside firewall (or whatever) to protect everything that isn't exposed.
- Users can't (in general) expose stuff by accident.

# Problems with Default Deny

- It stops business moving quickly
- Some activities are almost impossible, or result in perverse architectures that expose more than would otherwise be at risk (large databases). And it doesn't work for IoT applications (hence the messy use of brokers for Hive, Nest, etc).
- It can be very expensive to implement well, and has very nasty failure modes.
- In 2018, it is an open question how much protection the “inside” router actually provides anyway. It gets treated as an airgap, and it simply isn’t. The cult of the firewall is strong.

# How Default Deny Ends Up



# Benefits of Default Permit

- It's much more flexible, and new applications can brought into service very quickly
- Because all sensitive systems have to protect themselves, it doesn't matter (as much) if a nearby system is compromised.
- Administrative load is spread out over departments

# Problems with Default Permit

- All machines are exposed to substantial threats
- Lack of central policy means “weakest link in chain” is a problem
- Auditors and other assessors will be very, very nervous
- But in reality, many networks are becoming like this

# So increasingly...

- We use network-level security to reduce the level of “simple” attacks, perhaps without having a formal DMZ (either by intent or by accident).
  - The formal DMZ pattern is used to protect small islands of sensitive systems, running defined, relatively slow-changing workloads.
- We use more appropriate host-based security to protect information assets elsewhere.
- We use policy to make sure that every host is protected (in the way that we use talking to our children to make sure that they never misbehave). *Note: This is sarcasm on many levels.*
- On campus, you have islands of protected networking in amongst a sea of what is effectively the open Internet.
- It is instructive if you have the opportunity to observe the University’s contortions on this topic, 20 years after the fact.
- Think: “how sensitive is the stuff on the CEO’s laptop when he’s travelling? What firewall is it behind?”

# Transport Attacks

I.G.Batten@bham.ac.uk

# Attacks on UDP and TCP

- UDP: datagrams, no “connection” state
- TCP: reliable, sequenced data
- Both routinely passed through firewalls
- Both routinely attacked

# Filtering

- Simple filtering looks at the IP address and the port number in both the source and the destination.
- So if you only want to receive traffic to your DNS server from a particular network, you write something like:
  - **pass** traffic to port 53 from network 147.188.0.0/16
  - **block** all traffic to port 53 from elsewhere.
- Another example is “accept email only from my filtering service” or “accept remote logins only from my branch office”.

# Why doesn't this always work as well as you want?

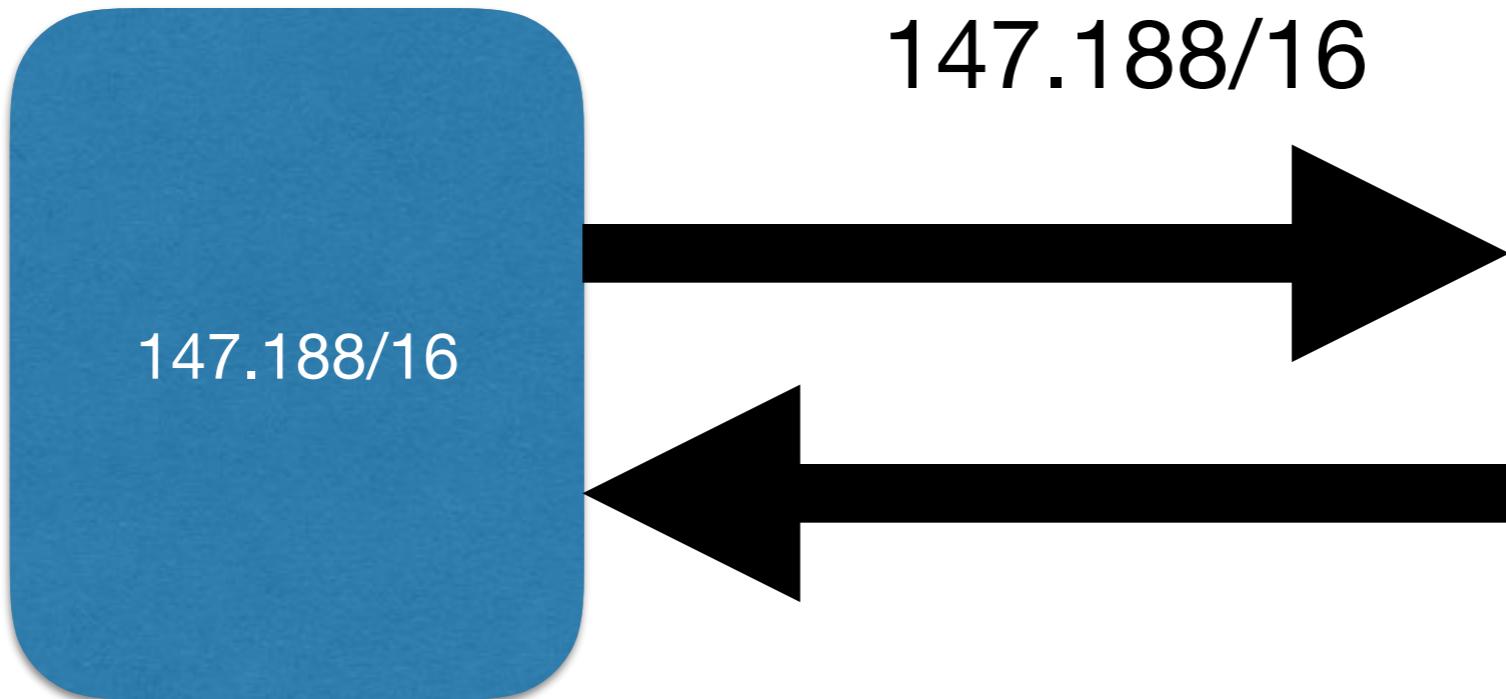
- The attacker is assumed to have full control over the entire contents of the packet.
  - This is a good assumption because it's true. Anything which assumes the attacker cannot set any byte of the packet to any chosen value is unlikely to be correct.
  - The attacker can make traffic appear to come from anywhere, including your “whitelisted” network.

# Seeing the responses

- It is trivial to forge packets coming from a particular address. Indeed, it's routinely done by accident.
- There are proposals to make it harder (BCP38) and RFCs discussing it (most recently RFC 6959) but most ISPs do not do effective egress control (“don’t let packets out with addresses I don’t control”) or effective ingress control (“don’t permit inbound packets with source addresses that are already inside my network”).
- It is in general harder to get hold of packets sent to an address you don’t control (requires alteration of routing tables, or strong interception capability). Different class of attacker.
- So interesting to see what can be done by an attacker who can send you arbitrary packets, but cannot see traffic not destined for them.

# What should be blocked, but isn't?

Discard all traffic  
outbound **not** from  
147.188/16



Discard all traffic  
inbound **from**  
147.188/16

This is all  
worth  
doing  
yourself,  
by the  
way

# What does this matter

- Attacker can bombard your UDP services with arbitrary traffic (but cannot see the responses)
- Attacker can bombard your TCP services with arbitrary traffic (but cannot see the responses)
- And other stuff, like ICMP, but that's much more likely to be filtered these days.

# UDP Attacks

- Can send packets in the hope of (for example) buffer overruns
- Can send packets in hope of DoS
- Can send packets to use services that are authenticated by source
- And can send packets as part of amplification attacks.

# Source Authentication

- One offender is syslog, UDP port 514.
- Send a packet to port 514, contents are appended to a log
- Sounds harmless, but swamps logging service so legitimate messages are dropped or ignored.
- Non-local syslog (eg, passing port 514 through a firewall from a remote site) is madness.

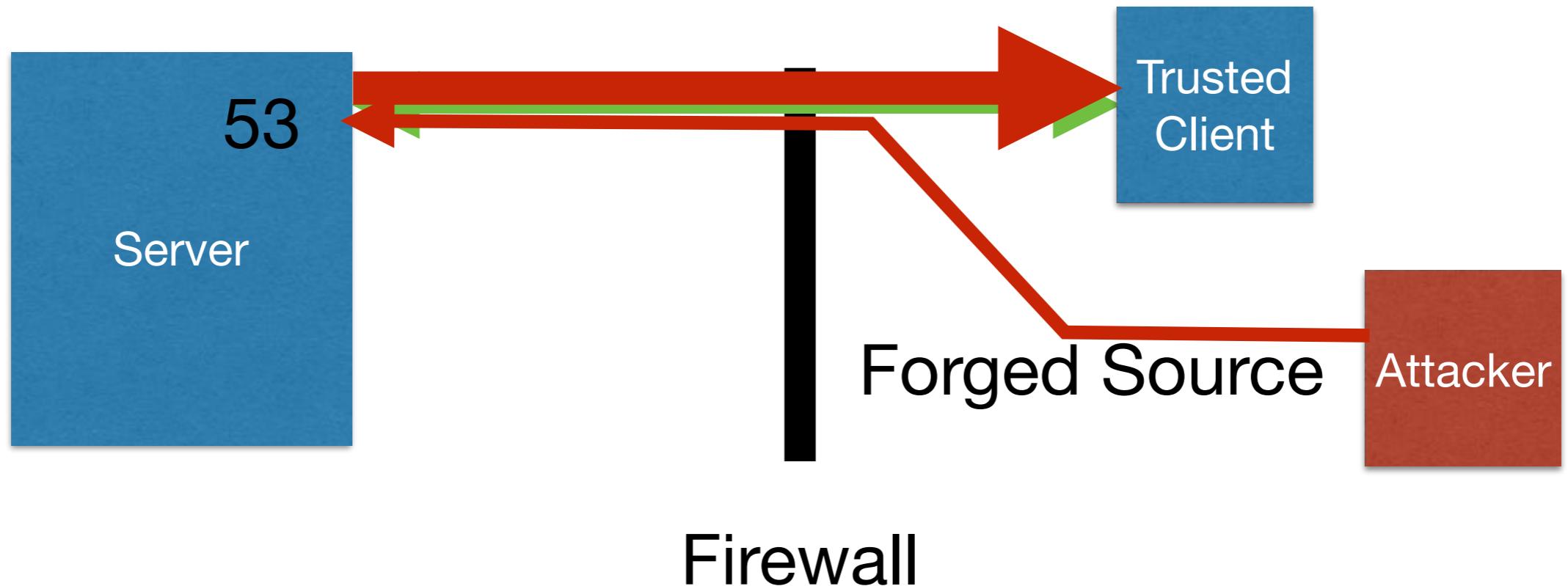
# Source Authentication

- Other serious offender (and it gets worse, see next page) are DNS, SNMP and NTP. In both cases, people restrict access to servers by source address because of concerns about information leakage, load on vital infrastructure and “pay for your own damned servers”. But...

# Amplification Attacks

- Nasty even without the amplification.
- Send a packet to an innocent intermediary, using a public service, with the source address spoofed to be a victim's address.
- Each packet elicits one or more response directed to the victim, coming from the intermediary.
- Some protocols (worst case was an NTP management protocol at around 1:10) amplify hugely.
- Original attacker cannot be traced without help of intermediary.

# Amplification Attacks



# Broadcast Attacks

- Wide range of attacks (“Smurf”, for example) available if you can send “directed broadcast” packets: packets which when they arrive on the final router are then broadcast, rather than unicast.
  - Send ICMP echo request packet to 147.188.0.0 with forged source address of 1.2.3.4, and in principle 1.2.3.4 will receive an ICMP echo response from every machine on campus.
- In practice, even the most dilatory firewall administrator blocks broadcast traffic.

# Splicing Attacks, etc

- TCP sequence numbers distinguish segments
- Need to be unique over long-ish periods of time to avoid reuse within segment lifetime (ie, if all connections started at 0, a flurry of short-lived connections between same ports on same machines would be problematic)
- TCP checksum is per-segment, so will not detect segments being swapped

# What if I can predict segment numbers?

- I can send a stream of packets which acknowledge data I can't see
- This sounds useless, but in fact is very, very useful

# TCP handshake

Client Address+Port	Client Address+Port
SYN	X

Anyone can send this

Server will send  
this to Client

Server Address+Port	Client Address+Port
SYN	Y
ACK	X+1

Client Address+Port	Server Address+Port
ACK	Y+1

Requires knowledge  
of Y

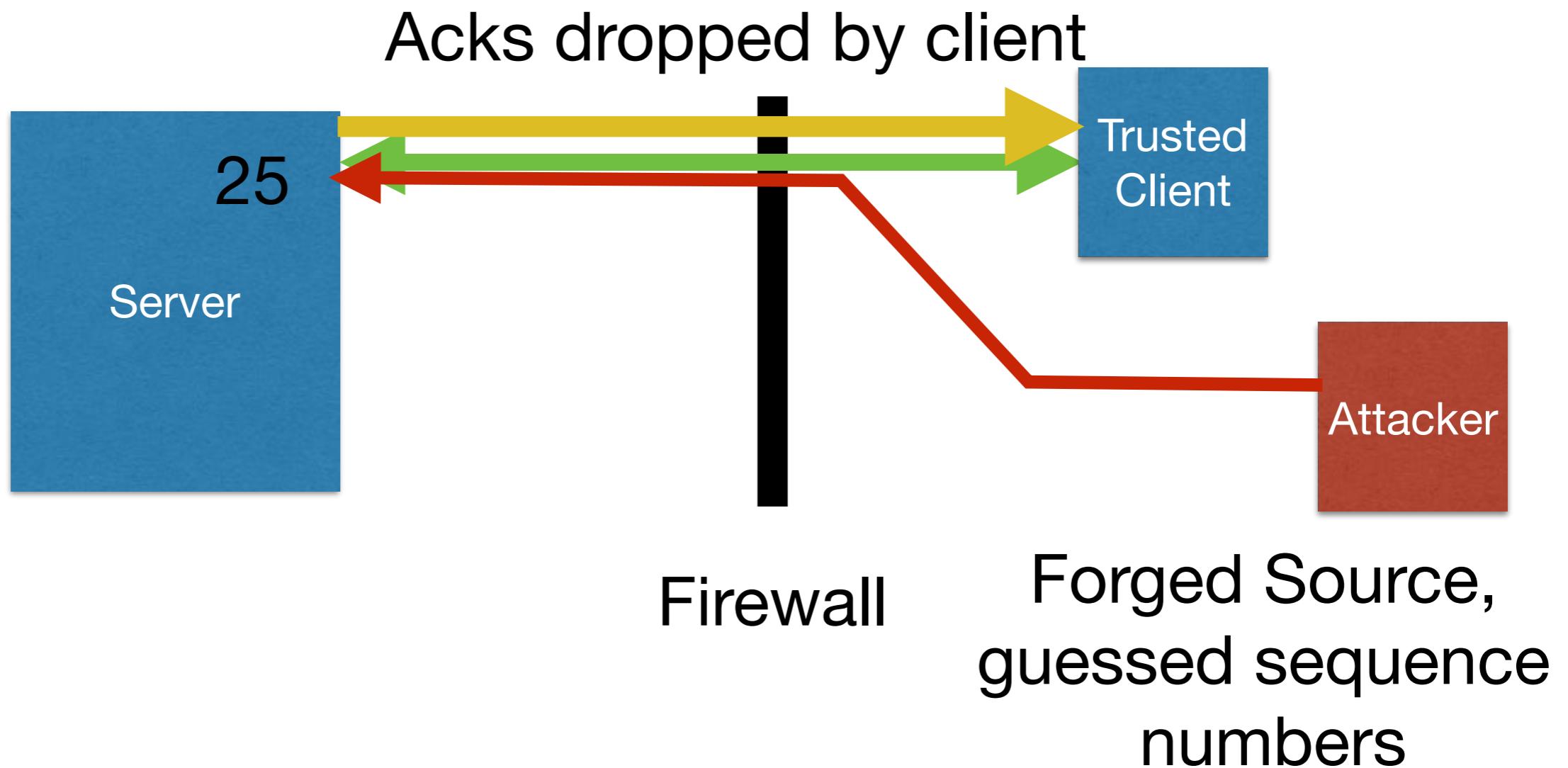
Subsequent traffic requires knowledge of X and Y

# Given X and Y, all sequence numbers can be predicted

```
i.g.batten@bham.ac.uk... Connecting to smart1.bham.ac.uk. via relay...
220 smart1.bham.ac.uk ESMTP Exim 4.84 Tue, 16 Jan 2018 10:24:26 +0000 71 bytes
>>> EHLO mail.batten.eu.org
250-smart1.bham.ac.uk Hello mail.batten.eu.org [147.188.192.250]
250-SIZE 104857600
250-8BITMIME          121 bytes
250-PIPELINING
250 HELP
>>> MAIL From:<igb@batten.eu.org> SIZE=26
250 0K          8 bytes
>>> RCPT To:<i.g.batten@bham.ac.uk>
>>> DATA
250 Accepted
354 Enter message, ending with "." on a line by itself
>>> .
250 0K id=1eb0Pq-0007zK-EJ
i.g.batten@bham.ac.uk... Sent (OK id=1eb0Pq-0007zK-EJ)
Closing connection to smart1.bham.ac.uk.
```

Even if I cannot see the bold text (Server→Client) I can often predict the length, and therefore send appropriate acknowledgements. If uncertain, “correct” acknowledgements will work, “incorrect” ones ignored.

# So



# What do I send?

- All packets forged Client→Real Server
  - SYN X
  - ACK Y+1 (having guessed Y)
  - EHLO mail.batten.eu.org\r\n, ACK Y+72 (71 bytes)
  - MAIL From:<igb@batten.eu.org> SIZE=26\r\n, ACK Y+193 (121 bytes)
  - RCPT To:<i.g.batten@bham.ac.uk>\r\n, ACK Y+201 (8 bytes)

# Relies on guessing ISN

- Turns out to be scarily easy
- Particularly when a machine is providing a public server you can connect to (http, DNS) and a private server available only to select clients (mail, telnet).
- Predicting ISN now given ISN a known time ago originally very easy (increments 128 per second and 64 per connection)
- Has got better, but not as better as it should...

# Phase Space Diagrams

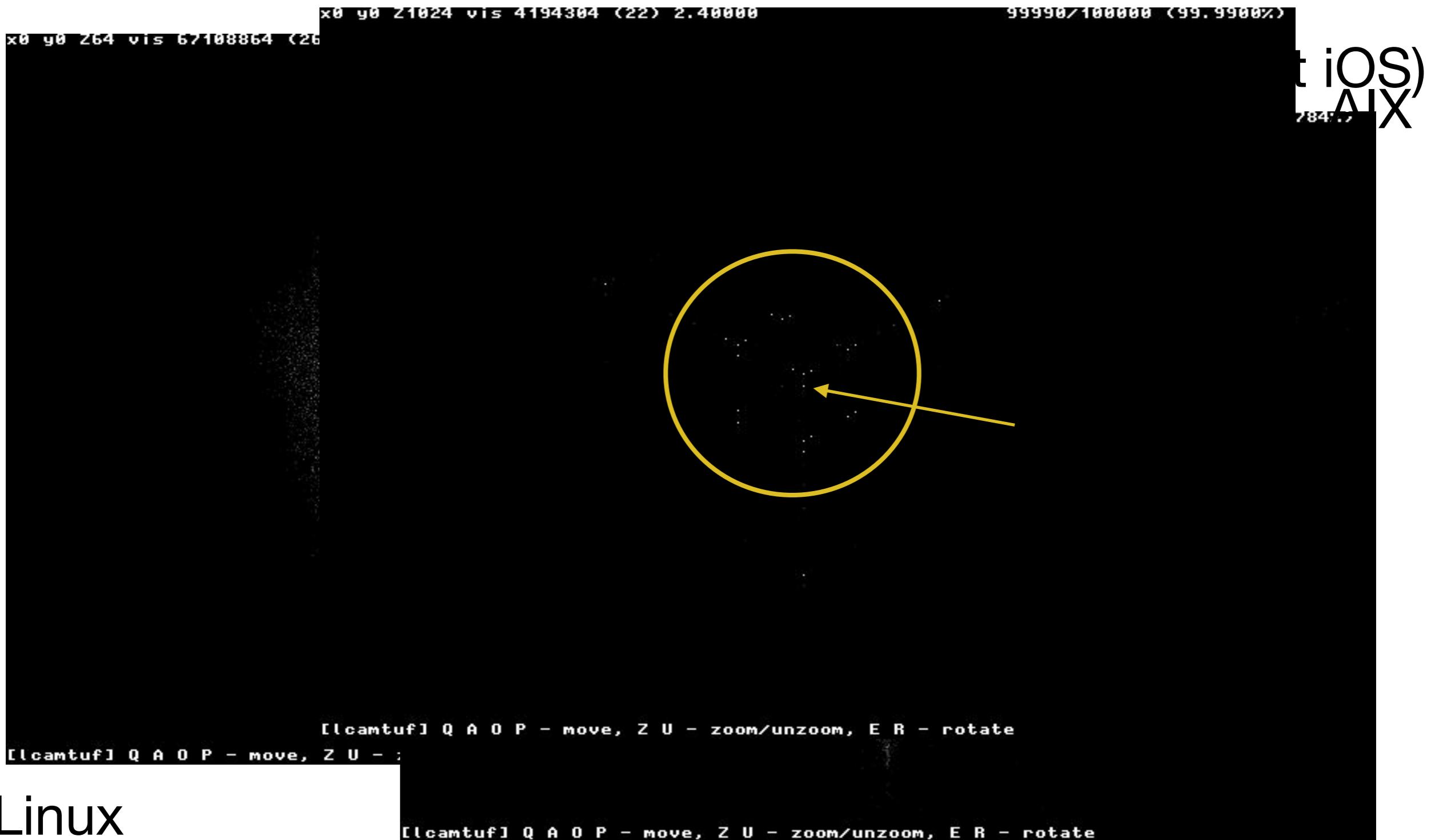
- <http://lcamtuf.coredump.cx/oldtcp/tcpseq/print.html>
- Generates points in three-space based on window of three values (which should yield a uniform cloud):

$$x[n] = s[n-2] - s[n-3]$$

$$y[n] = s[n-1] - s[n-2]$$

$$z[n] = s[n] - s[n-1]$$

# But real world not very random



# Remember...

- I don't need to succeed every time, or anything like it.
- Broken attempts to connect probably won't appear in any operating-system logs
- The example I have shown are old, but there are lots of old systems in real networks.
- pf firewall (and commercial products like Cisco ASA) will now replace sequence numbers with good randoms, on a network-wide basis. How? Well, one way follows...

# Handshake offload

- Attacker can “SYN Flood” servers by just sending lots of SYNs, and then not responding further.
- For public services, attacker can insert a random source address into each request: hard to filter, hard to track.
- Modern OS should survive, but it’s surprisingly unpleasant: most mitigations slow legitimate traffic as well. Less modern OSes, who knows?
- Solution: do the three-way handshake in the firewall.

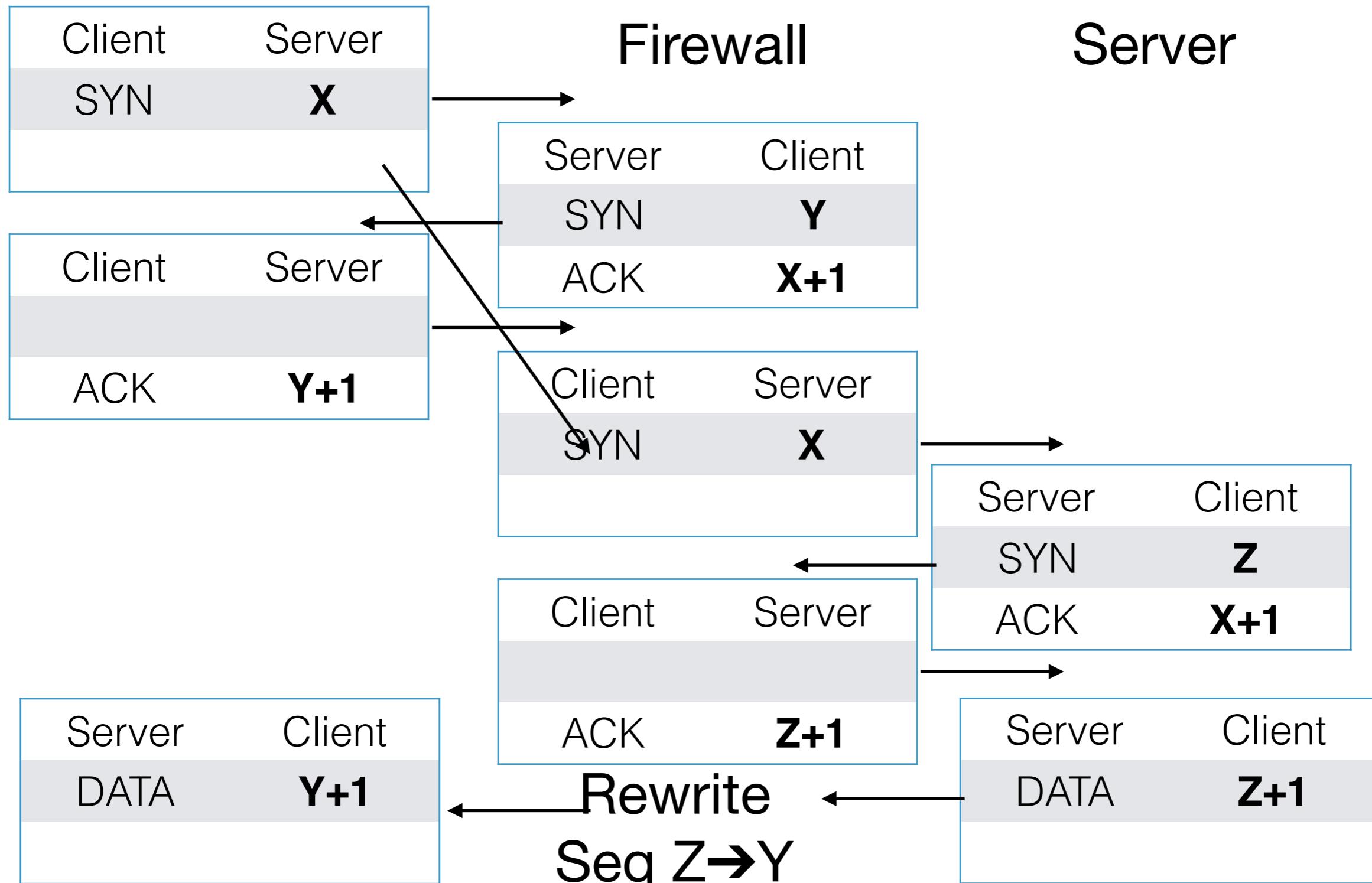
# Handshake offload

- How can a firewall deal with the three-way handshake?
- Client sends SYN
- **Firewall** sends SYN/ACK, with Firewall-chosen ISN
- **Firewall** waits for ACK from Client
- **Firewall** now passes initial SYN to server, waits for SYN/ACK with server-chosen ISN.
- **Firewall** calculates offset between ISN it chose and ISN the server chose, and applies that to all subsequent packets on the connection.

# Handshake offload

Client

Server



# Why do this?

- Modern OSes should be able to cope with a “SYN Flood”
- But there are lots of legitimate (ish) reasons why you might be running old software which can’t.
  - Being held to ancient OS ransom by application vendors and your own developers is a thing.
- Border protection is wise.

# Handshake offload also...

- Rewrites the sequence number to guard against it being guessed, because the firewall can use a strong RNG.
- BOGOF: Two protections for the price of one.
-

# Other TCP Entertainment

- Send SYN (possibly using forged source) and then do nothing: consumes resources.
- Send RST (possibly using forged sources) to break connections
- The list is endless
- In short, punching small holes in firewalls to permit selected TCP connections through doesn't work in 2018.

# Tasks for a firewall

- Drop packets from “outside” which have “inside” IP numbers
- Drop packets from “outside” which have “odd” IP numbers (RFC1918, loopback)
- Drop packets from “inside” whose source is not an inside network
- Rate limit stuff susceptible to abuse
- If possible, do SYN/SYN-ACK/ACK in the firewall
- If possible, re-randomise ISN
- If possible, deal with fragments, dropped packets, etc.
- Expose the services you need

# Network Security 3: Hosts

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Hosts hold Assets

- So we need to make sure that only authorised users can access the hosts, and that those authorised users can only do what they are authorised to do.
- And we need to have a way to see when bad things happen, and to investigate when bad things do happen.

# Log Files

- Most sites do not do logging properly.
- Important that logs be:
  - complete
  - accurate
  - trustworthy
  - secure

# Don't Filter!

- Disk space is incredibly cheap
- Log files may be useful in a forensic situation, and the most trivial stuff may matter
- Save everything you can, and worry about filtering to make it easy to read, not to make the volumes smaller

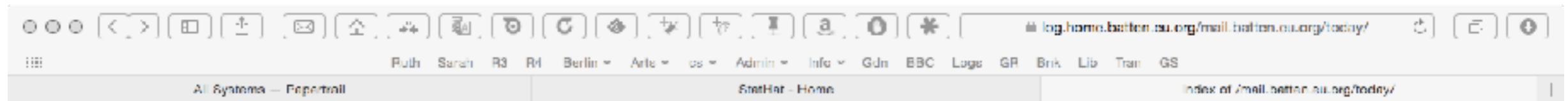
# Get it away!

- An attacker can manipulate local log files
- An attack may find it harder to manipulate remote log files, especially if the machine does nothing but accept log entries (papertrail, for example).
- Some people even run the most vital logging straight to a printer in the data centre

# Practical Logging

- Let's look at home-brew (syslog-ng) and commercial (papertrailapp).

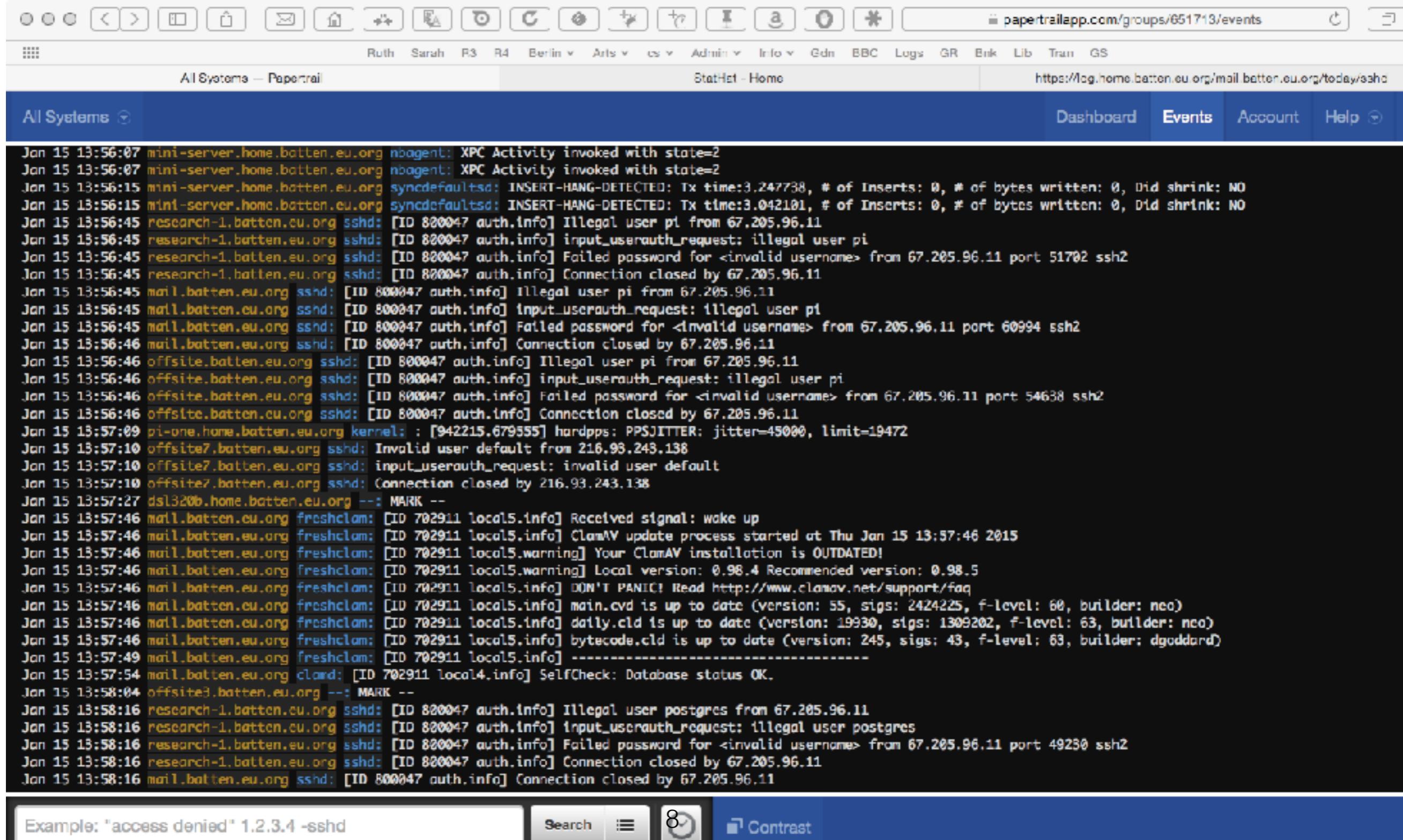
# syslog-ng



## Index of /mail.batten.eu.org/today/

..		
<a href="#">annotate-milter</a>	15-Jan-2015 13:25	20K
<a href="#">clamd</a>	15-Jan-2015 13:28	7231
<a href="#">cti_cymruadb</a>	15-Jan-2015 13:35	18K
<a href="#">cvs_expires</a>	15-Jan-2015 01:00	1201
<a href="#">freshclam</a>	15-Jan-2015 13:27	33K
<a href="#">imap</a>	15-Jan-2015 13:29	121K
<a href="#">imaps</a>	15-Jan-2015 13:25	107K
<a href="#">lmtpunix</a>	15-Jan-2015 13:31	26K
<a href="#">opendkim</a>	15-Jan-2015 13:31	18K
<a href="#">sendmail</a>	15-Jan-2015 13:35	146K
<a href="#">squatter</a>	15-Jan-2015 03:32	2005
<a href="#">sshd</a>	15-Jan-2015 13:40	178K
<a href="#">sync_client</a>	15-Jan-2015 13:40	15K
<a href="#">tel_prune</a>	15-Jan-2015 02:30	120

**papertrail.com**



# Patching

- Vulnerabilities are found all the time
- Often in security-sensitive code (they're less interesting in other code, after all, so people aren't looking for them)
- That means regular patching of “exposed” machines
  - If the company policies make this hard, because of configuration management, fix the policies

# Linux, for example

- Weekly report:

```
Reading package lists...
```

```
Reading package lists...
```

```
Building dependency tree...
```

```
Reading state information...
```

```
The following packages have been kept back:
```

```
  fake-hwclock
```

```
The following packages will be upgraded:
```

```
  curl file libcurl3 libcurl3-gnutls libcurl4-openssl-dev libevent-2.0-5
```

```
  libmagic1 python-rpi.gpio python3-rpi.gpio
```

```
9 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

# Problems with patching

- You have code which relies on a bug, and the patch fixes a bug and breaks code
- The patch itself is broken
- Patching doesn't work correctly and breaks the machine

# The solution is...patching

- The solution to problems with patching is to patch more often
  - Problems rapidly show themselves
  - Each patch set is smaller
- Alternative is old machines that everyone is frightened of and don't dare update

# Custom Software

- A typical problem for Linux is needing magic kernels (eg “server” options on a “desktop” system).
  - NO\_HZ\_COMMON y
  - HZ\_PERIODIC n -> y
  - NO\_HZ y -> n
  - NO\_HZ\_IDLE y -> n
  - +NTP\_PPS y
- The solution is to build a process for compiling kernels from new source with the required options.

```
Linux pi-one 3.12.36+ #1 PREEMPT-igb Sun Jan 18 19:35:05 GMT 2015 armv6l GNU/Linux
Linux pi-two 3.12.36+ #737 PREEMPT Wed Jan 14 19:40:07 GMT 2015 armv6l GNU/Linux
Linux pi-three 3.12.36+ #737 PREEMPT Wed Jan 14 19:40:07 GMT 2015 armv6l GNU/Linux
```

# Service Minimisation

- There is no point in running services you do not need
- Attackers can attack anything that is running; they do not helpfully avoid trying the things you've forgotten are there
- So you have the problem of running machines with the bare minimum of services.

# Why minimise?

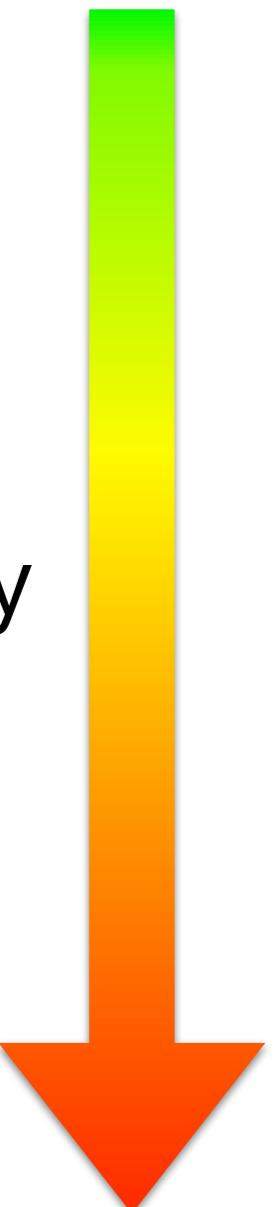
- Services, in this context, are pieces of software accessible to a user who is not logged on.
  - “Accessible” means “can have bytes sent to”: it may not matter if they have not authenticated.
- Every service has security problems, either known or unknown.
- Reducing the number of services reduces the “attack surface”: the number of places an attacker can try.

# Services

- SMTP listener for incoming email
- IMAP listener for mail storage
- SSH listener for login
- MySQL/Postgres/etc listener for databases
- DNS listener for nameservers
- Etc

# Options

- Don't install the service
- Don't run the service at all
- Run the service, but have it only available locally
- Run the service, but protect it from the network
- Run the service, exposed to the network



# Don't install the service / Don't run the service

- If a machine is not a fileserver, you don't need the fileserving software
- If a machine is not a webserver, you don't need the web service
- Don't install it

# Run locally

- Sometimes you need service X to support service Y
- An example would be needing MySQL or some other database to support a network management product
- The database is only needed by local services, so can listen on a local socket (AF\_UNIX, “named pipe”) or on 127.0.0.1.

# Protect from network

- Sometimes this means running a service that is only needed locally, but where the software always wants to listen more widely. Use a firewall.
- Alternatively, if it only needs to listen to **some** sources, don't listen more widely:
  - syslog servers should only listen to machines they are logging for
  - May require VLANs

# Run publicly

- If you must. If you must.

# Tools to find services

- nmap and other port mapping tools
- netstat -a and other “what’s listening” tools
- lsof/pfiles to look at what a process is listening on
- ps -ef to look at processes (ps -ax on palaeolithic Unixes)
- Windows equivalents are available

# Example

```
[igb@offsite7 ~]$ netstat -a | grep LISTEN
tcp      0      0 offsite7.batten.eu.o:domain *:*
                                         LISTEN
tcp      0      0 localhost.localdomain:domain *:*
                                         LISTEN
tcp      0      0 *:ssh                *:*
                                         LISTEN
tcp      0      0 localhost.localdomain:rndc *:*
                                         LISTEN
tcp      0      0 localhost.localdomain:smux *:*
                                         LISTEN
tcp      0      0 *:mysql              *:*
                                         LISTEN
tcp      0      0 localhost.localdomain:http *:*
                                         LISTEN
tcp      0      0 *:domain             *:*
                                         LISTEN
tcp      0      0 *:ssh                *:*
                                         LISTEN
tcp      0      0 *:https              *:*
                                         LISTEN
unix  2      [ ACC ]    STREAM      LISTENING   2151442158 //var/syslog-ng.ctl
unix  2      [ ACC ]    STREAM      LISTENING   2151441266 @/com/ubuntu/upstart
unix  2      [ ACC ]    STREAM      LISTENING   2151442725 /var/lib/mysql/mysql.sock
unix  2      [ ACC ]    STREAM      LISTENING   2151442135 /dev/log
[igb@offsite7 ~]$ sudo lsof | grep :mysql
mysqld  787  mysql  10u    IPv4          2151442724      0t0        TCP *:mysql (LISTEN)
[igb@offsite7 ~]$ ps -ef | grep 787
mysql  787  6 Jan11 ?  14:04:53 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql
--user=mysql --log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/
mysql.sock
igb     9522  9490  0 20:53 pts/0    00:00:00 grep 787
[igb@offsite7 ~]$
```

# Example

```
[igb@offsite7 ~]$ netstat -a | grep LISTEN
tcp      0      0 offsite7.batten.eu.o:domain *:*
tcp      0      0 localhost.localdomain:domain *:*
tcp      0      0 *:ssh          *:*
tcp      0      0 localhost.localdomain:rndc *:*
tcp      0      0 localhost.localdomain:smux *:*
tcp      0      0 *:mysql        *:*
tcp      0      0 localhost.localdomain:http *:*
tcp      0      0 *:domain       *:*
tcp      0      0 *:ssh          *:*
tcp      0      0 *:https         *:*
unix    2      [ ACC ]      STREAM      LISTENING    2151442158 //var/syslog-ng.ctl
unix    2      [ ACC ]      STREAM      LISTENING    2151441266 @/com/ubuntu/upstart
unix    2      [ ACC ]      STREAM      LISTENING    2151442725 /var/lib/mysql/mysql.sock
unix    2      [ ACC ]      STREAM      LISTENING    2151442135 /dev/log
[igb@offsite7 ~]$ sudo lsof | grep :mysql
mysqld  787  mysql  10u    IPv4      2151442724      0t0      TCP *:mysql (LISTEN)
[igb@offsite7 ~]$ ps -ef | grep 787
mysql   787  6 Jan11 ?  14:04:53 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql
--user=mysql --log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/
mysql.sock
igb     9522  9490  0 20:53 pts/0    00:00:00 grep 787
[igb@offsite7 ~]$
```

```
ians-macbook-air:clocks igb$ telnet offsite7 mysql
Trying 64.188.45.237...
Connected to offsite7.batten.eu.org.
Escape character is '^]'.
4
5.1.73(5rbq4NCVg%~o3MhFeI
```

# Example

```
ians-macbook-air:clocks igb$ telnet offsite7 mysql
Trying 64.188.45.237...
Connected to offsite7.batten.eu.org.
Escape character is '^]'.
4
5.1.73(5rbq4NCVg%~o3MhFeI
```

# Example

Why are we running mysql? This machine is running cacti, a network monitoring package that needs a mysql data, but is itself accessed only over https.

```
ians-macbook-air:clocks igb$ telnet offsite7 mysql
Trying 64.188.45.237...
Connected to offsite7.batten.eu.org.
Escape character is '^]'.
4
5.1.73(5rbq4NCVg%~o3MhFeI
```

# Example

Why are we running mysql? This machine is running cacti, a network monitoring package that needs a mysql data, but is itself accessed only over https.

```
ians-macbook-air:clocks igb$ telnet offsite7 mysql
Trying 64.188.45.237...
Connected to offsite7.batten.eu.org.
Escape character is '^]'.
4
5.1.73(5rbq4NCVg%~o3MhFeI
```

add **bind-address = 127.0.0.1** to /etc/my.cnf and restart

# Example

Why are we running mysql? This machine is running cacti, a network monitoring package that needs a mysql data, but is itself accessed only over https.

```
ians-macbook-air:clocks igb$ telnet offsite7 mysql
Trying 64.188.45.237...
Connected to offsite7.batten.eu.org.
Escape character is '^)'.
4
5.1.73(5rbq4NCVg%~o3MhFeI
```

add **bind-address = 127.0.0.1** to /etc/my.cnf and restart

```
[igb@offsite7 ~]$ sudo vi /etc/my.cnf
[igb@offsite7 ~]$ sudo service mysqld restart
Stopping mysqld: [OK]
Starting mysqld: [OK]
[igb@offsite7 ~]$
```

# Example (cont'd)

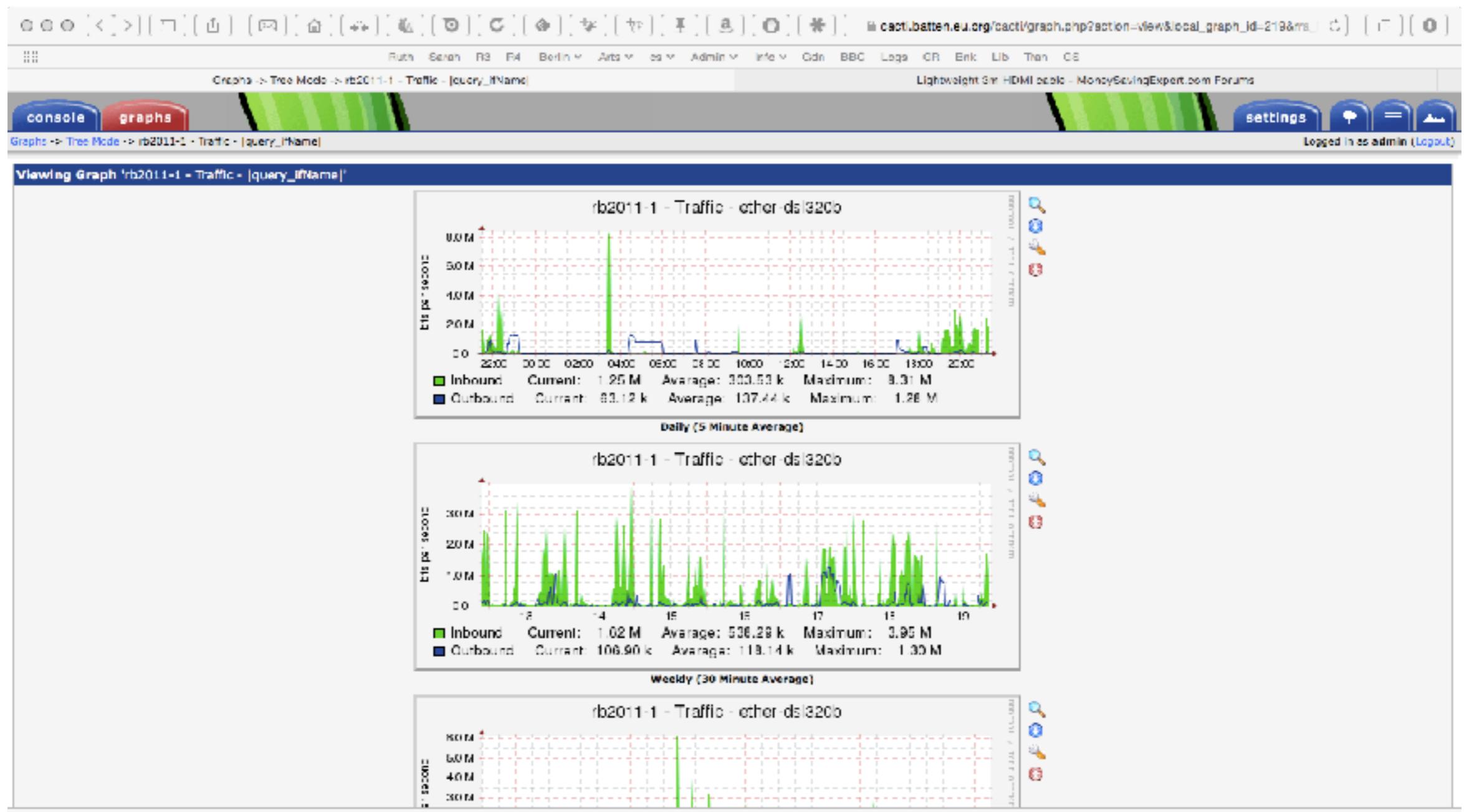
```
[igb@offsite7 ~]$ ps -ef | grep mysql
root      9765      1  0 21:00 pts/0    00:00:00 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --socket=/var/lib/mysql/mysql.sock --pid-file=/var/run/mysqld/mysqld.pid --basedir=/usr --user=mysql
mysql     9870  9765  0 21:00 pts/0    00:00:00 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
--log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mysql.sock
igb      9914  9490  0 21:01 pts/0    00:00:00 grep mysql
[igb@offsite7 ~]$ sudo lsof -p 9870 | grep TCP
mysqld  9870 mysql  10u  IPv4          2350892539      0t0        TCP localhost.localdomain:mysql (LISTEN)
[igb@offsite7 ~]$ netstat -a | grep mysql
tcp        0      0 localhost.localdomain:mysql *:*                  LISTEN
unix  2      [ ACC ]         STREAM     LISTENING      2350892540 /var/lib/mysql/mysql.sock
[igb@offsite7 ~]$
```

# Example (cont'd)

```
[igb@offsite7 ~]$ ps -ef | grep mysql
root      9765      1  0 21:00 pts/0    00:00:00 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --socket=/var/lib/mysql/mysql.sock --pid-file=/var/run/mysqld/mysqld.pid --basedir=/usr --user=mysql
mysql     9870  9765  0 21:00 pts/0    00:00:00 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
--log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mysql.sock
igb      9914  9490  0 21:01 pts/0    00:00:00 grep mysql
[igb@offsite7 ~]$ sudo lsof -p 9870 | grep TCP
mysqld  9870 mysql  10u  IPv4          2350892539      0t0        TCP localhost.localdomain:mysql (LISTEN)
[igb@offsite7 ~]$ netstat -a | grep mysql
tcp        0      0 localhost.localdomain:mysql *:*                  LISTEN
unix  2      [ ACC ]         STREAM     LISTENING      2350892540 /var/lib/mysql/mysql.sock
[igb@offsite7 ~]$
```

```
ians-macbook-air:~ igb$ telnet offsite7.batten.eu.org mysql
Trying 64.188.45.237...
telnet: connect to address 64.188.45.237: Connection refused
Trying 2607:f2e0:10f:14:4321:4321:5e6:9ee6...
telnet: connect to address 2607:f2e0:10f:14:4321:4321:5e6:9ee6: Connection refused
telnet: Unable to connect to remote host
ians-macbook-air:~ igb$
```

# And to test...



# Summary

- We are running cacti, a service which needs mysql
- Mysql was installed so it was listening to the internet generally, but was only needed locally
- Mysql modified to only listen locally
- Cacti still works
- This is a real example: I found it while doing the slides!

# Network Security 4: Isolation

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Last Lecture

- Reducing services to minimum
- Removing services which are not being used
- Restricting access to services which are needed to support other services

# Least Privilege

- Old-style, the easiest way to run system services is as root / Administrator
- Any compromised service therefore has access to the whole machine
- First step: run services as unprivileged users, with appropriate magic to do anything that needs privilege in a secure way.

# Why do you need root?

- To open privileged ports
  - Some operating systems allow you to do this without privilege, or with fine-grained privilege
  - Alternatively, open the port, then permanently rescind your privilege
- To write user files
  - There are other, better ways to do this
  - So whole “cyrus” mail system runs as user cyrus

# The next step: isolation

- Processes running on a machine can “see” each other, and exploit weaknesses up to attaching a debugger.
- So an attacker who compromises one service can attempt to gain control of other services.
- Eventually they may be able escalate to root via a chain of progressively more “powerful” vulnerabilities, and the processes they take over may also be useful in their own right

# In passing...

- In 2018, we don't care quite as much about protecting operating system images, so the “unprivileged” service is probably more important than the operating system instance.
- Real data is more important than keeping the machine running, as we can rebuild from media or restart from snapshots.
- The security model of the operating is precisely the wrong way around: it worries about the precious binary of /bin/cat far more than about your final year project.

# So ideally...

- You only have one service per machine (where “machine” means “operating system instance”, as we will see).
  - Then, an attacker who controls one service just controls that service, and there’s nothing else of value on the machine: doesn’t prevent the attack, but mitigates its consequences and avoids escalation.
- But machines are expensive, in terms of power, space, administrator effort, etc.
- Operating system licenses and support contracts aren’t necessarily free, either.

# Virtualisation and other OS isolation facilities

- Virtualisation covers a wide range of techniques for running **guest** operating systems, or things that look like operating systems, in parallel with other (**host**) operating systems.
- Unprivileged instructions are metal, privileged instructions which interact with real hardware are trapped and emulated
- It's worth looking at the spectrum of options.
  - There is some hand-waving in this, as the distinctions may not be quite as clear-cut as I imply

Got here 1500 Tue 16

# Spectrum of Isolation

- Type 1 Hypervisors
- Type 2 Hypervisors
- Paravirtualisation
- Jails / Zones / “Containers”

Ordering is up  
for debate

Lighter than  
virtualisation,  
but same  
purpose

Ease of Deployment

Degree of Isolation

# Type 1 Hypervisors

- Host runs a hypervisor, which does nothing but manage guest operating systems. You can't run anything in the host, and it's normally protected from the outside world.
  - Hush, hush, whisper it: the Hypervisor is often actually Linux...
- Guest operating systems are unmodified, and think they are running on a defined and supported hardware platform.
- Offers a wide range of facilities for redundancy, load balancing, migration, etc, etc.
- With minor exceptions, each OS instance runs its own networking, talking to virtual bridges or virtual VLAN, often with hardware assistance (VT-c, SR-IOV)
- VMware's big enterprise solution, ideal for big server farms. In best practice, the hypervisor has its own network interfaces on a private management network.

# Type 2 Hypervisors

- Full-fat operating system running on the host
- Which then supports a hypervisor which allows guests (again unmodified) to run
- Guests can (for example) display in a host-OS window.
- Guest OSes might have a bridge to the network, but might be NAT-ed and therefore all traffic goes through the (full-fat) host OS networking stack.
- Full-size host OS is attractive target for attackers: an attacker who is root on the host (via attacks on its processes) can almost certainly immediately compromise all guests.
- VMware Fusion, Virtualbox, etc.

# “Real” Virtualisation

- Guest and host can be completely disjoint: common scenario is running Windows as a guest in a Linux host, or vice versa. Great for developers, great for PaaS, great for data centre managers (most computers are idle most of the time, so better value per U of rack space)
- Resource allocation and scheduling can be complex and difficult to control, but are not a topic for this course.

# Para Virtualisation

- Operating system is slightly modified, rather than running unchanged, so that it uses special network, graphics and other drivers, which exist only as stubs in the guest OS, and often a lot of reliance on the host virtual memory subsystem.
- Arguably more efficient, but requires the OS vendor to co-operate.
- Hypervisor has more visibility of guest, and potentially vice versa (active research in department on running virus/malware scanners in host to look “into” guest OSes: security as a service).
- Usually Linux or other open-source OS as the guest: Xen-guest versions of other OSes are intermittently available (Solaris was for a while, MSFT research did a Xen-XP in the early days) but are not common.

# Zones, Jails, Containers

- “Jails” are available in FreeBSD.
- “Zones” (aka “Containers” in marketing speak) are available in Solaris derivatives (ie, commercially and supported).
- Machine runs one kernel, but a subset of processes are isolated with their own init, filesystem, sometimes networking.
- Very lightweight and easy to manage, consumes almost no additional resources. Only needs one OS license.
- Security properties look good, but fair to say analysis is complex as the interface between container and its host is “fat”. Ripe topic for a PhD. Cambridge Mirage project (in OCAML!) is an interesting alternative spin to look at.
- Linux analogues came later but are storming up the inside into the lead: Linux Containers now (because of Docker) do almost everything that Solaris container do, and are cheaper/easier/more common.
  - I believe you can’t have separate IPsec configurations in distinct containers on Linux. About three people in the world care about this, one of them me, and the security properties of it would be tricky to assess.

Hardware

Host OS (if Type 2 Hypervisor)

Hypervisor

Solaris

Linux

FreeBSD

Zone

Zone

Zone

Container

Container

Container

Jail

Jail

Jail

Collectively probably managed with Docker today

# chroot ()

- Very old-fashioned Unix tool, the ultimate basis for things like zones and containers.
- After chroot (“/some/directory”), all processes which are started from point on see “/some/directory” as “/”. Processes cannot get outside that part of the filesystem.
- Specifically, “cd ..” stops at the new “/”, similarly “cd /”.
- Complex to make user code work correctly (libraries, devices, /proc, etc).
- Anyone clever enough to use it is clever enough to use something better.

# How to choose?

- All forms of virtualisation separate the number of application instances from the volume of tin in the machine room (“tin” — computers for the (un) hip).
- The lightest forms separate the number of application instances from the number of OS instances.
- All offer some sort of mobility from “shut down, copy the underlying files, start up on another machine” to almost real-time failover.

# “Real” Virtualisation

- Doesn’t require modified Operating Systems
  - Probably only realistic choice for running Windows on multi-tenant platforms and getting any support, although situation is changing rapidly.
  - Ideal for running legacy OSes on modern hardware (DOS 5.0! Windows XP! BeOS!)
  - Some configurations allow you to hide legacy networking code behind modern stack, but security properties of this again hard to evaluate.
- Usually has extensive support for redundancy, relocation, live upgrade, etc.
- Hardware required to do it well is expensive
  - Requires lots of RAM, in particular
  - Requires high-end SAN storage behind the servers to support mobility.
    - Storing the virtual images on NAS is for the brave

# Para Virtualisation

- Good luck getting OS support!
- Ideal for selling private servers to people that want them.
- Guests are reliant on host for some services (for example, clock synchronisation is usually done by host OS)
- Not so easy for live mobility (because the “state” of the guest is in part the “state” of the host) although a lot of work has been done.
- Good enough for many, perhaps most, purposes: Amazon, etc.

# Zones, etc

- Fully supported by vendors, indeed encouraged by vendors.
- Doesn't always help you with different OSes, but Docker.
- Might help with different OS versions (Solaris 11 can run Solaris 10 and Solaris 8 applications in zones, although it isn't bug-for-bug compatible as it's "really" a Solaris 11 kernel).
- Mobility is usually limited to moving cold systems, but there is active work on improving this.
- Cheap, lightweight, ideal if your stack fits it.
- Docker is changing the trade-offs as we speak.
- I think in 2017 a LAMP/etc application which isn't deployed via Docker should be asked "why not?"

# HMG Footnote

- *I have a medical textbook which has special “except if you’re military and in the field” sections denoted with camouflage borders to the pages. I need something similar for “what about classified systems?”*
- It is highly unlikely that any of these isolation techniques are regarded as strong enough to separate different classifications, and unlikely that they are regarded as enough to separate different assets at the same classification at the higher levels without a lot of effort and the use of assured (ie, expensive) solutions.
- The segregation is just too hard to audit, either at the time of selection or on an on-going basis.

# Network Virtualisation

- Virtualisation suites allow you to build virtual networks inside the virtualised environment, and run virtual firewalls and virtual intrusion detection systems.
- We'll talk about this after we've talked about firewalls and IDSes.

# Network Security 6: Defence in Depth

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Recap

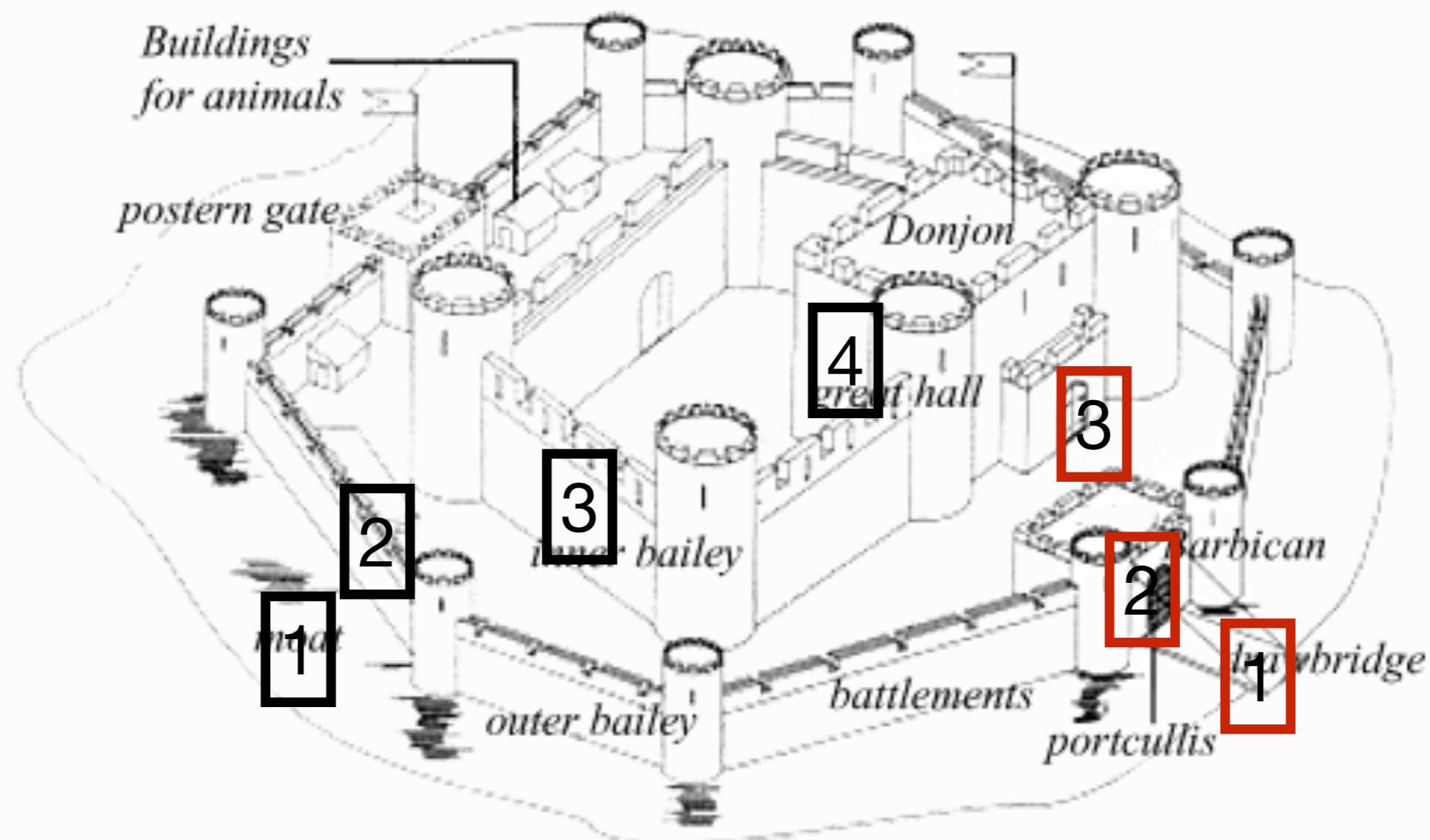
- Logging tells us what happened, and might allow us to spot patterns early
- Patched systems have fewer security problems
- Services that aren't running can't be broken into
- Services which aren't listening are hard to break into

# Defence in Depth

- Idea is that multiple defences add (multiply?) together to improve security
- If one defence is breached, there are others still standing
- Model has a long history...

# Castles

Diagram of a medieval castle



# Problems

- Assumption that defences are independent
  - Not just physically or logically, but in terms of tools needed to break them
  - Attacker who can knock down one wall can knock down others
  - Attacker who can brute-force one encryption key and brute-force others

And can be counter-intuitive: which is safer?

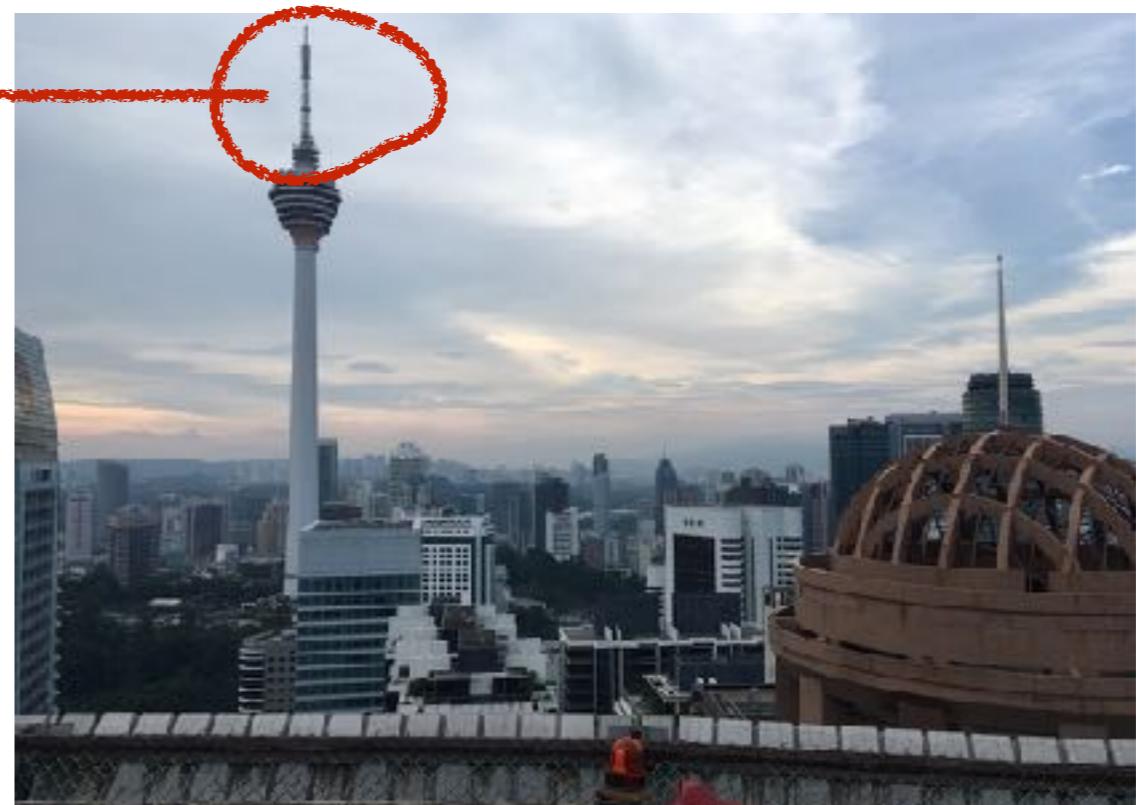
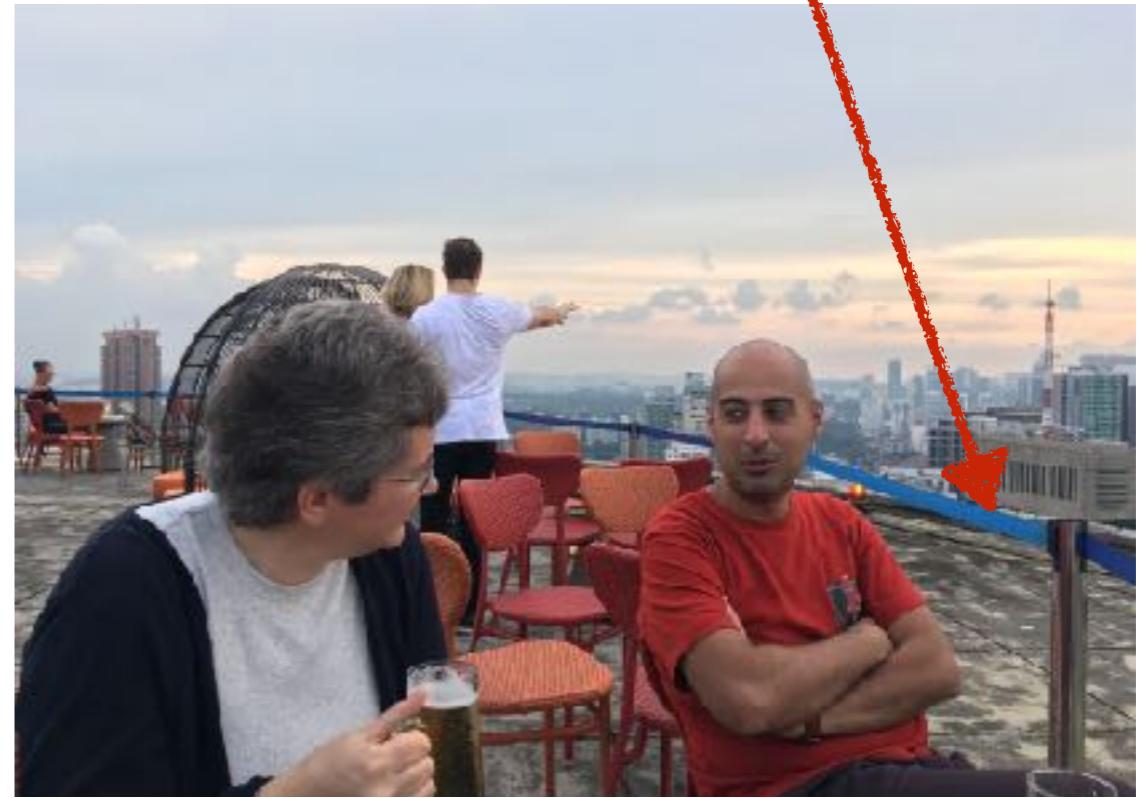


# When do planes crash?

- Takeoff is the riskiest phase of flight
- An aircraft accelerates down the runway until it reaches speed  $V_1$ , at which point it cannot stop before the end of the runway.
- It needs to get to  $V_2$  (usually greater than  $V_1$ , except on ten mile runways), which is minimum safe climb-out speed.
- At some point (usually between  $V_1$  and  $V_2$ ) it will reach  $V_R$ , at which point it “rotates” (starts to take off).
- Engine failure between  $V_1$  and  $V_2$  is very dangerous
- Certification rules are “must be able to get from  $V_1$  to  $V_2$  with one engine failed”. Twins therefore have 200% power installed, Fours 133%.

# Twin v Four not obvious

- Twin will crash on takeoff or goaround if two engines fail (assuming 1% chance of engine failing)
  - $(0.01 + 0.01) * 0.01 = 0.0002$
- Four will crash on takeoff or goaround if two engines fail
  - $(0.01 + 0.01 + 0.01 + 0.01) * (0.01 + 0.01 + 0.01) = 0.0012$
  - $0.0012 = 6 * 0.0002$
  - **Six times greater risk** (assuming risk of engine failure is constant per-engine, which is true to a first approximation).



# Reality much more complex

- Reality is much more complex, and complicated by the certification regimes:
  - Twin-engined aircraft over water (ETOPS) are subject to much stricter rules than fours
  - On the other hand, fours can cruise on two engines and sometimes land. Risk of 3 independent engines failing 8x less than two out of two failing (probably less: calculation is complex as some two engine scenarios are unflyable)
- But having 200% of minimum take off power is preferable to having 133%

# Independence

- But if a plane runs out of fuel, or enters a cloud of volcanic dust, all the engines fail, whether there are one, two, three, four or eight engines



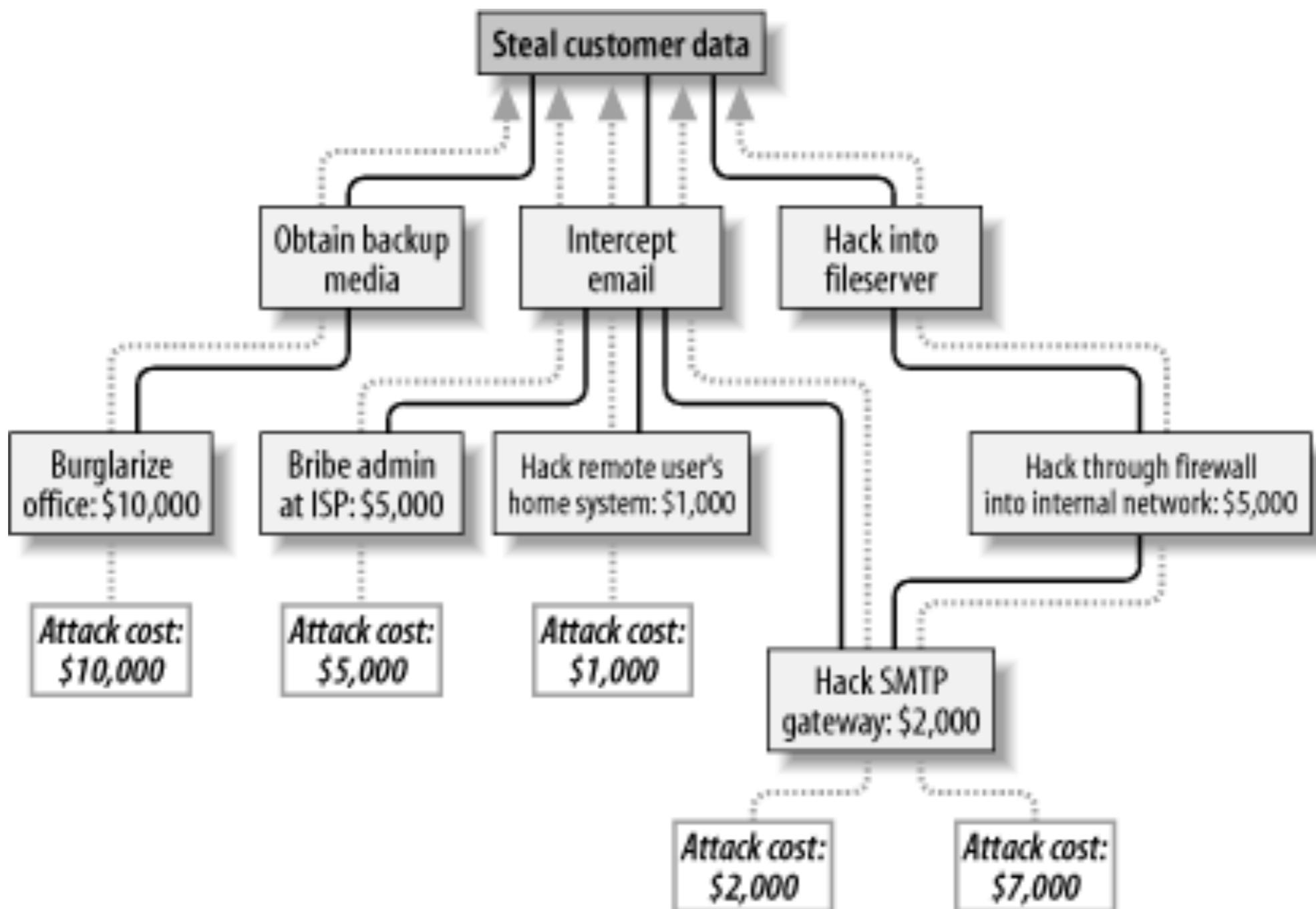
# For Network Security

- It is very tempting to think that having lots of defences equates to having defence in depth.
- But one strong lock is preferable to ten weak locks, as an attacker who can break a weak lock can break ten.
  - And a door with ten locks is weakened by ten sets of holes drilled in it.
- We need to make sure we are getting increased protection.

# Attack Trees

- Build a tree, with the attacker's goal at the top, and the various ways he might achieve that descending from it.

# Example



# For each risk, controls

- Backups can be encrypted
- Hackable systems can be made less hackable
- Bribeable staff can be vetted, their jobs divided in two, etc.

# Building attack trees is hard

- There is research work both on building them and on analysing them (ripe field for PhD).
- Looking for independence is hard, too

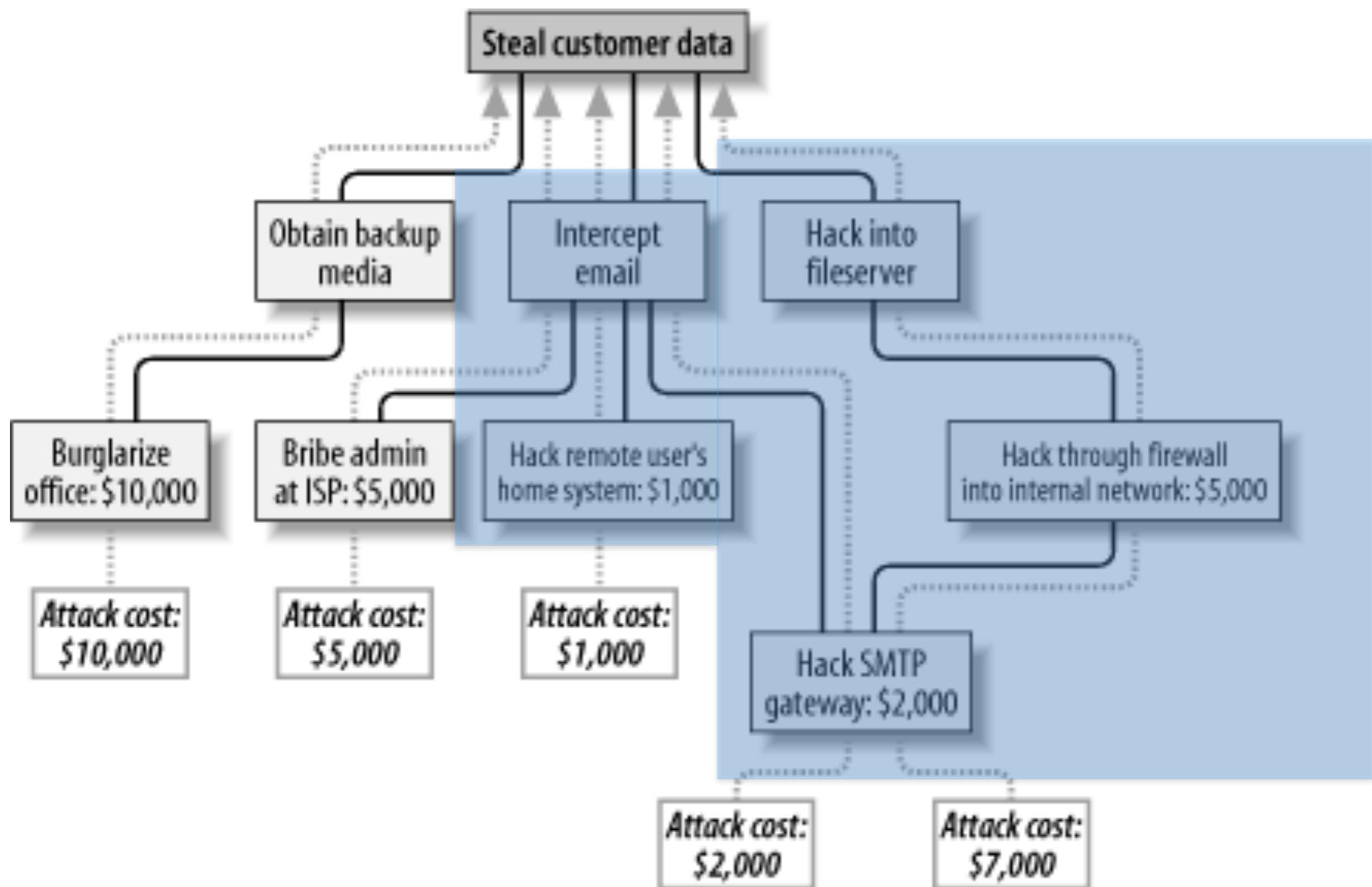
# Exercise

- Take five minutes, and build an attack tree for changing your end of year marks on a marks database.
- Assume the database is on a computer, which is behind a firewall, which is administered by an administrator and used by lecturer
- Be imaginative

# Depth?

- Tempting to think that a system protected by two firewalls (or whatever) both of which need to be hacked is more secure than a system protected by one.

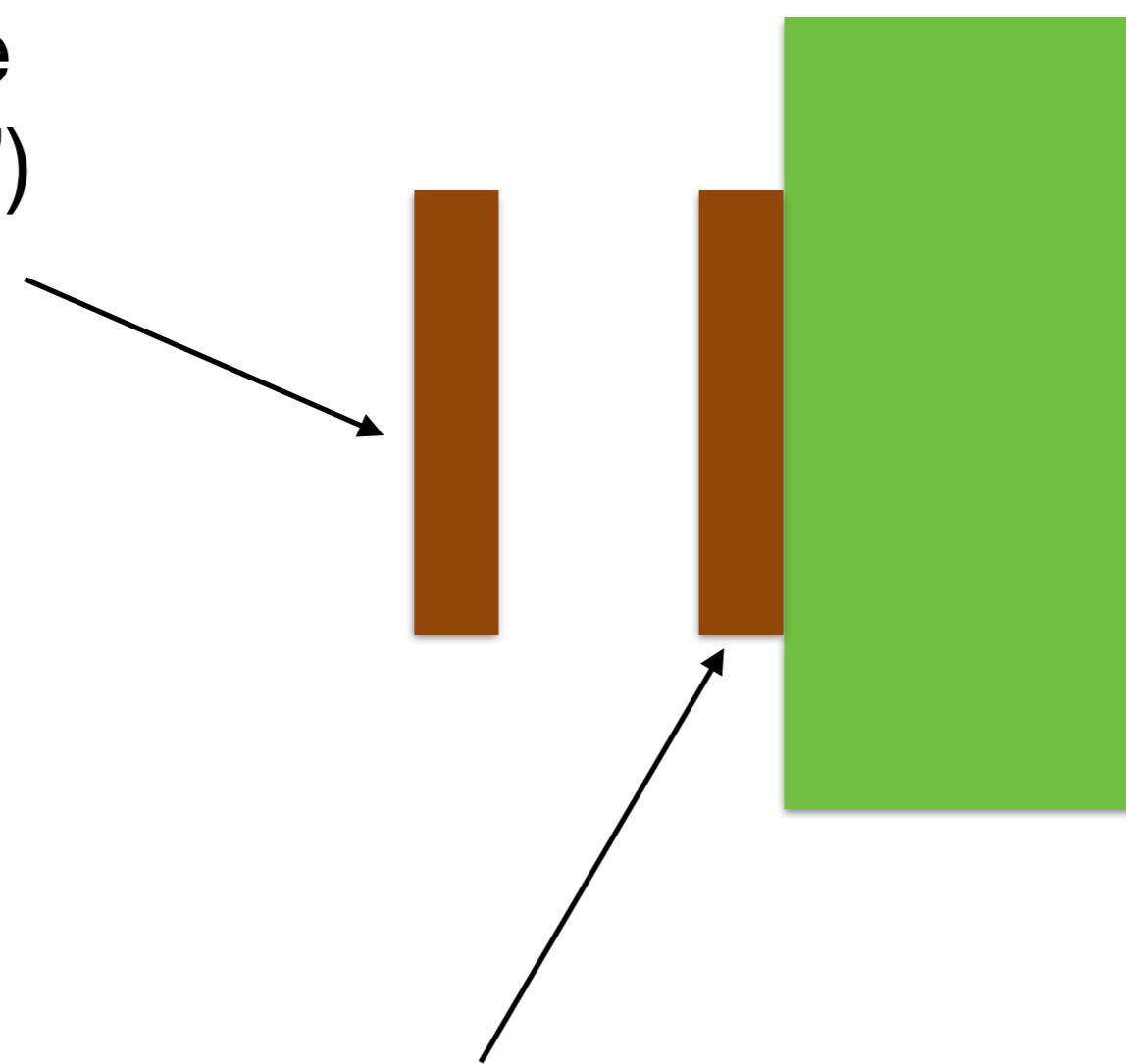
# Are these independent?



Standalone  
("hardware")  
Firewall

Computer

Operating System Firewall



# Firewalls and Attack Surfaces

- You can filter packets using a firewall on a computer
- You can configure the services on that computer securely
- But an attacker who has a get-root privilege escalation attack can bypass both

# Firewalls and Attack Surfaces

- You can filter packets using a firewall in a separate box, placed in front of a server
- However, if there is an authentication server which permits users to log in to the firewall and to the computer, an attacker who can attack the authentication server is admin on the firewall and the computer.

# Complexity is hard

- In general, the more devices and elements are involved in a security solution, the less likely that it is accurately analysed
- Tradeoff between simplicity (and therefore ease of analysis) and defence in depth is not one that can be given a general answer.

# Subsidiary Protocols

- Attack trees often miss attacks on “subsidiary protocols”.
- Classic example is DNS.
- Suppose I configure some access control mechanism (Apache .htaccess) to permit access to a sensitive document from secure.bigcorp.com.

# DNS PTR records

- How do you find out the name of a machine from an IP number?
- For address 1.2.3.4, you form this name:
  - 4.3.2.1.in-addr.arpa
- And you look up the PTR record for that name.
- The 3.2.1.in-addr.arpa “zone” is controlled by the owner of 1.2.3.0/24.
- So they can add this record to the DNS.

# Solution

- Attacker controls 3.2.1.in-addr.arpa, but does not control bigcorp.com.
- Solution is to follow up any query IP->name by looking up the name and checking that the IP number is one of the address records.

# Example

```
ians-macbook-air:~ igb$ nsupdate -k update-key
> server offsite7.batten.eu.org
> update add 215.150.187.81.in-addr.arpa. 86400 in ptr gromit.cs.bham.ac.uk
>
> ians-macbook-air:~ igb$
```

```
ians-macbook-air:~ igb$ dig -x 81.187.150.215

; <>> DiG 9.8.3-P1 <>> -x 81.187.150.215
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4764
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;215.150.187.81.in-addr.arpa. IN PTR

;; ANSWER SECTION:
215.150.187.81.in-addr.arpa. 86400 INPTR      gromit.cs.bham.ac.uk.

;; Query time: 107 msec
;; SERVER: 147.188.244.250#53(147.188.244.250)
;; WHEN: Tue Jan 27 14:26:00 2015
;; MSG SIZE  rcvd: 79

ians-macbook-air:~ igb$
```

```
ians-macbook-air:~ igb$ dig gromit.cs.bham.ac.uk

; <>> DiG 9.8.3-P1 <>> gromit.cs.bham.ac.uk
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32162
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;gromit.cs.bham.ac.uk.      IN A

;; ANSWER SECTION:
gromit.cs.bham.ac.uk.  85439 IN A
;; Query time: 6 msec
;; SERVER: 147.188.244.250#53(147.188.244.250)
;; WHEN: Tue Jan 27 14:27:02 2015
;; MSG SIZE  rcvd: 54

ians-macbook-air:~ igb$
```

# Other Protocols

- NTP can be abused (but rarely is): if you really need accurate time and rely on it, then you need your own reference clock
- DHCP and other configuration protocols can be abused, but it is difficult to do remotely

# Network Security 6: Network Elements

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Spectrum of boxes

- Hubs
- Simple Bridges
- Switches
- Filtering Switches
- Routers
- Filtering Routers/Firewalls (simple firewalls)
- Stateful Firewalls (mainstream firewalls)
- Deep Packet Inspecting Firewall
- (IDS/IPS/spooky things)

More protection, more  
inspection, more issues



# Hubs

- Copy ethernet packets from one interface to all interfaces, probably just electrically (ie, don't look at the packets at all).
- Provide no security, and are (let's hope!) being taken out of networks.



# Simple Bridges

- Copy ethernet packets from one interface to the others, dropping packets that are malformed or corrupt.
- Provide almost no protection, aside from extremely simplistic flooding attacks not seen since the 1980s.



# Filtering Bridges and Switches

- Copy ethernet packets from one interface to the other, if the bridge believes that the source and destination are on different sides of the bridge.
- Prevents an attacker on network A from observing traffic between two hosts on network B.
- Switches stop hosts from seeing anyone else's traffic (although not hard to bypass)



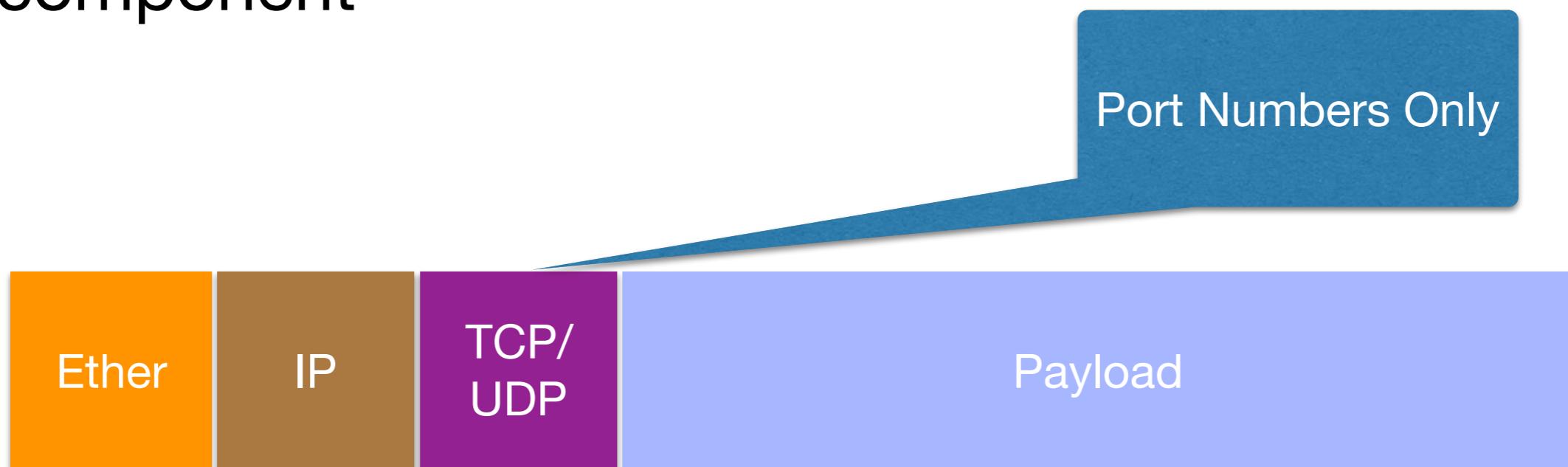
# Routers

- Looks at the IP header and chooses an interface to send the packet out of
- Blocks some malformed packets, and has similar filtering effects to a switch.
- Doesn't (in general) propagate broadcast packets



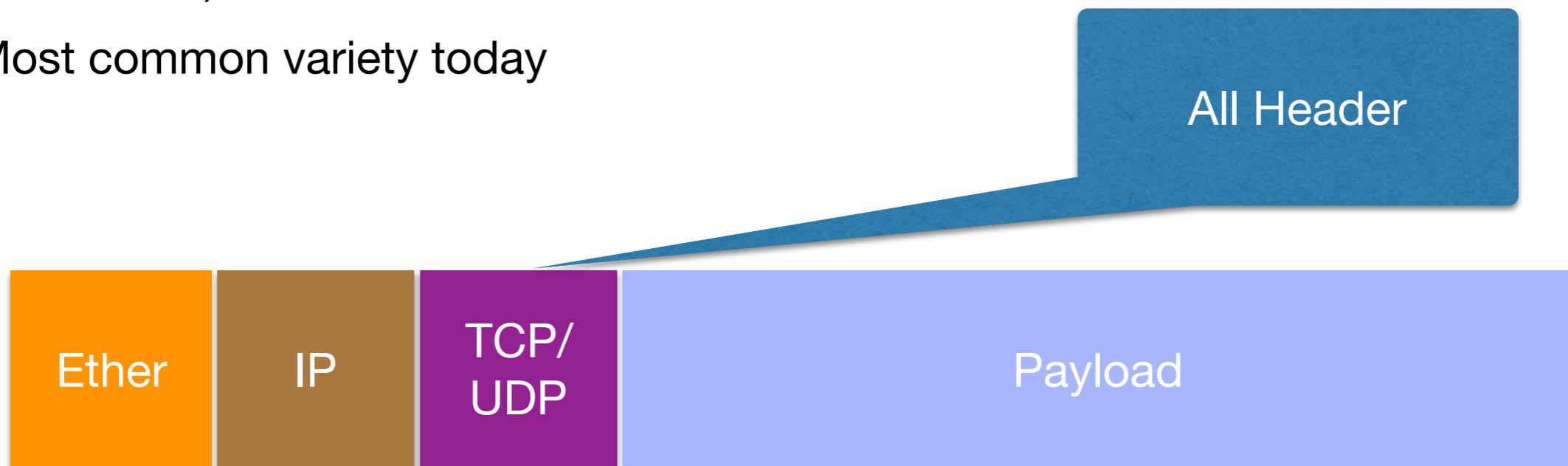
# Simple Firewalls

- Act as a router, but also look at TCP and UDP port numbers and block/pass based on those values
- Will shield services on systems from an attacker
- The minimum requirement to be a security component



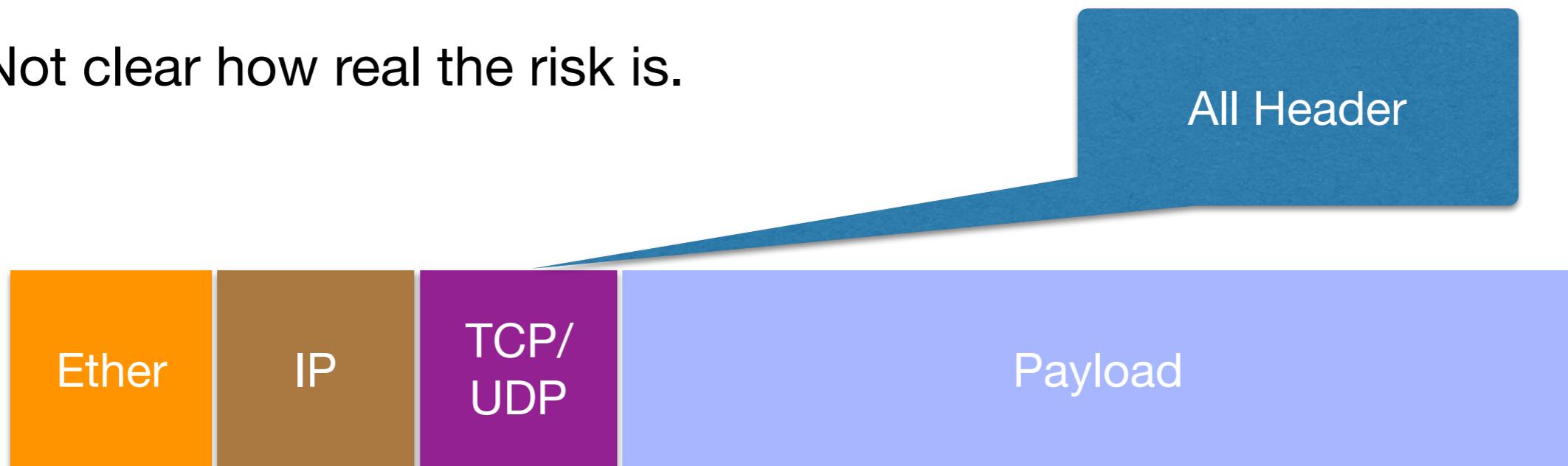
# Stateful Firewalls

- Look not only at port numbers, but the whole state of the TCP connections through a router.
  - Tracks the connections, so it knows that SYN-ACK or RST are the only sensible responses to SYN
  - Tracks sequence numbers, to ensure the next packet is actually a sensible packet for the connection
- Not only shields services, but blocks more complex attacks on TCP connections (next week!)
- Most common variety today



# Stateful Firewalls++

- There are some firewalls (Cisco ASA, for example, and the FreeBSD pf suite) which can rewrite TCP sequence numbers, reassemble fragments and re-order packets.
- Attacks on sequence numbers will come next week
- Idea is that protected systems only ever execute the most well tested code paths, so attacker cannot probe weakness in less well tested code. Makes all TCP stream canonical.
- Not clear how real the risk is.



# Deep Packet Inspection

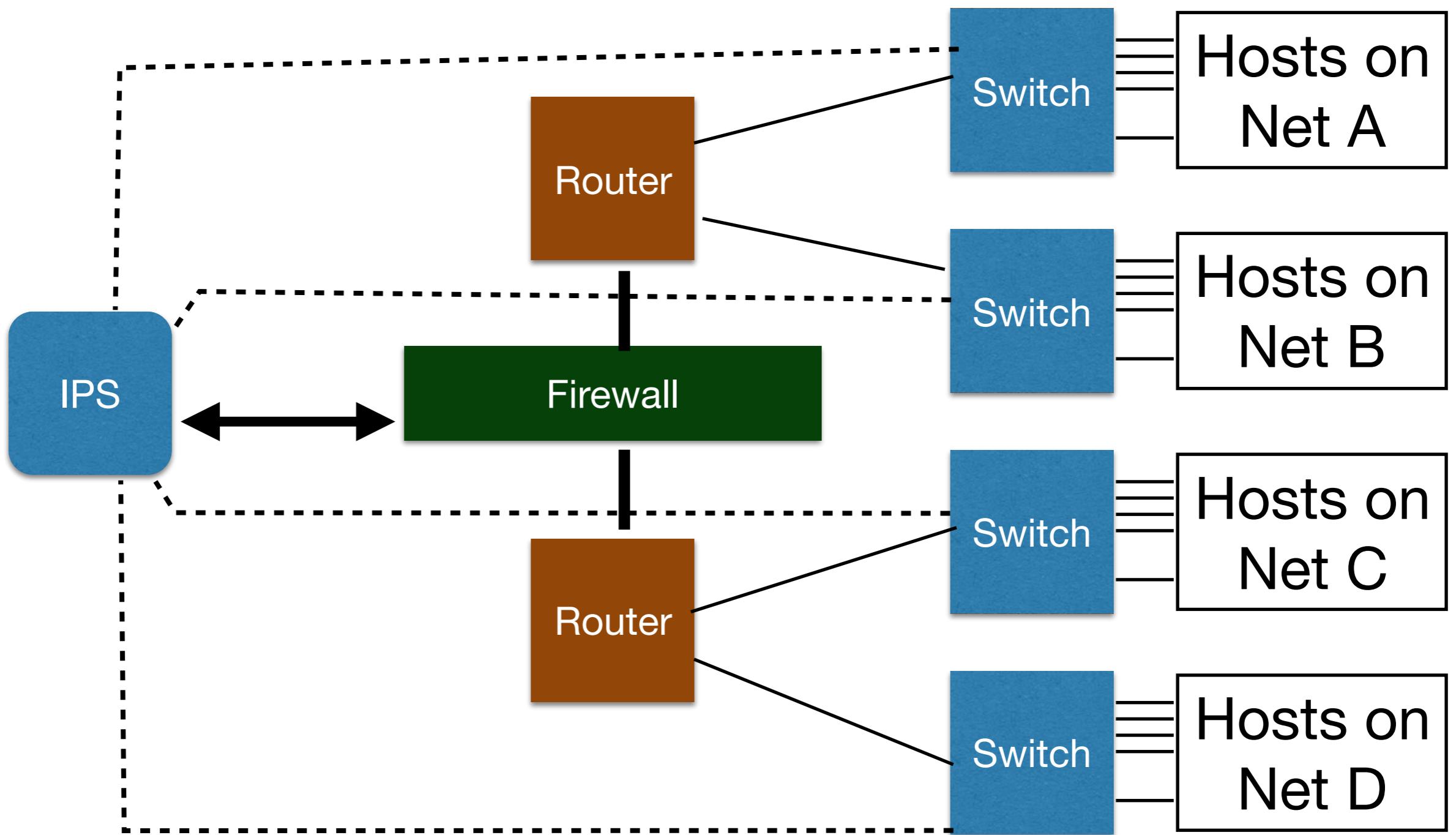
- Looks at entire packet as it passes through router
- Can operate as a virus scanner, can check protocols aren't being abused, can perform arbitrary re-writes.
- Can even block encryption it doesn't know about by considering randomness, although that is an arms race.
- Sometimes regarded as unethical when deployed in company networks, arguably regulated in public networks (Phorm debacle).



# IDS/IPS

- Intrusion Detection / Prevention Systems
- Receives copy of traffic from switches and routers
- Examines whole packet to looks for attacks and other worrying conditions
- Either reports them (D) or instructs routers and firewalls to drop traffic (P).

# IDS/IPS



# So what can we do with the components?

- We want to detect threats,
- block threats,
- respond to threats

# Switches and Bridges

- MAC-based filtering is about local traffic, as all remote packets will have local MAC addresses from the last router they went through
- However, can be useful to stop attacks from insiders, and to avoid common configuration mistakes.

# Switch filtering

- Although it is very labour intensive, access ports can be filtered to only accept traffic from the expected device.
  - Prevents MAC spoofing and addition of cascaded switches and hubs to add unauthorised devices
  - Doesn't scale well
  - Also worth doing in the vicinity of routers to make “whoops, I plugged the wrong cable” errors less fatal

# Switch Authentication

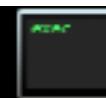
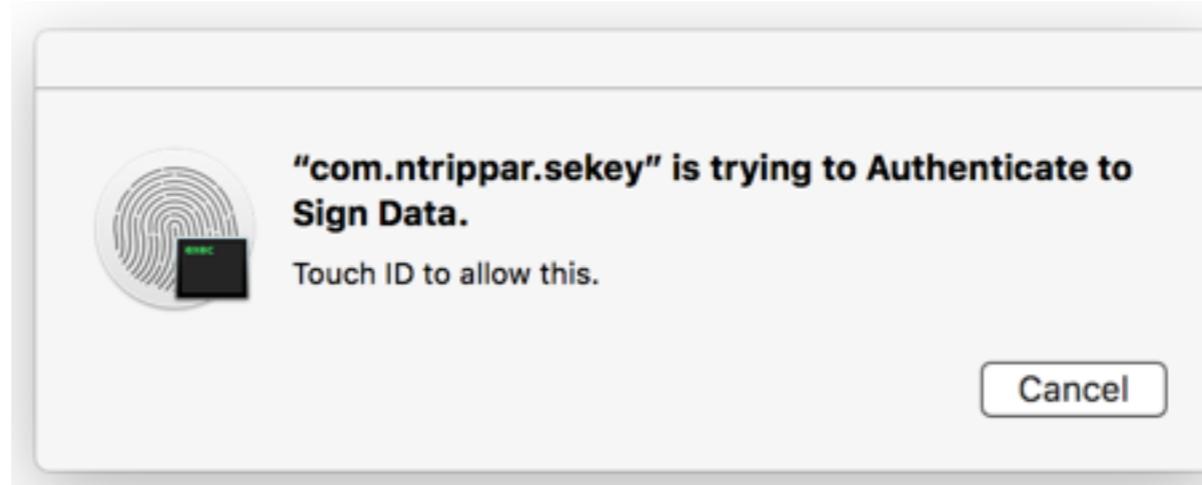
- You can automate switch filtering with 802.1x authentication: devices are authenticated to the switch before they can send traffic, somewhat in the manner of wireless authentication.
  - Also “posture checking” on things like virus scanners and patch levels
- Very fashionable in about 2005–10, lots of kit sold, but not clear how many deployments succeeded.

# Problems with 802.1x

- Basic idea is sound: when a port sees a new device, it requests authentication before enabling the port, passing the MAC address, allowing the MAC address onto a designated VLAN, etc.
- Problems arise with:
  - Cascaded switches (which one do you authenticate to?) (“multiple supplicant”)
  - Failure of radius/etc server is potentially devastating
  - Printers and other “non-user” devices: where do the keys go? Securely? TPM offered hope, as does Trustzone/SGX/SEP but little enthusiasm from suppliers.
  - Machines that need to boot unattended, ditto.
  - If doing posture analysis, do you really have the agent for all your equipment? No? So how do you know which devices should be running it and which can’t.
  - If he’s here, Darren will now tell us his war stories...

# In passing, SEP...

```
debug1: service_accept received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Offering public key: ECDSA SHA256:WGE2TJwCPhAKMxr+Ildcoqg1jD5iwm/B8kSZ6o
Z+n6s ecdsa-sha2-nistp256
debug1: Server accepts key: pkalg ecdsa-sha2-nistp256 blen 104
```



com.ntrippar.sekey Touch ID to Authenticate to Sign Data →

# Routers

- Prevent attacks involving sending packets that are not IP: routers will not pass ARP, weird non-IP protocols (assuming router is not configured to route them), etc.
- On their own, routers do not provide any filtering

# Simple Firewalls

- Can be part of a router or part of an end system (and a “router” may be a computer with two interfaces).
- Conceptually usually thought of as being “in front” of the network interface, but in reality is usually part of the first code run when a packet arrives (for ingress filtering) or the last code run when a packet leaves (for egress filtering) (more complex on Linux)

# Simple Firewalls

- Allows the blocking of everything apart from certain protocols and ports, or...
- ...the passing of everything except certain protocols and ports.
- Cannot check legality of TCP packets at more than the most basic level (packets in isolation, rather than as part of a flow)
- Struggles to filter and check UDP.
- Struggles in the face of NAT
- Unless you are using an ancient Cisco router, unlikely to be used for any security application today

# Stateful Firewalls

- Inspects all the elements in the packet header, and keeps track of all connections through the firewall
- Also called, unsurprisingly, “connection tracking” (also “keep state”, from the configuration option in the BSD firewall code).
- Code is closely related to NAT (my memory is that it predated NAT, which leveraged the code, but I could be wrong).
- Can check that TCP packets are in the expected state, as we will explore next week.
- Match UDP packets heading out and coming back in, to permit services to cross the boundary.
- Performance implications in complex or busy networks, and require careful engineering to avoid intentional and unintentional denial of service attacks.
- Failover and standby configurations notably tricky, especially in the face of asymmetric routing
- Most common form of firewall today

# Statefull++

- (This is my terminology).
- As well as checking that packets are expected, modifies them to make them “more” expected: removes duplicates, waits for retransmissions, reassembles fragments.
- Protects against unknown risks in rare code paths; packets only arrive in common states and conditions
- Also means that hardware assist and “fast path” is always used, with rare/untested “slow path” not needed
  - You have to worry about the test coverage in TCP stacks.
- Understands HTTP, SMTP, FTP and so on and similarly enforces “sensible” rather than “strict” limits on command length, syntax, etc.
- Basic idea is that your ASA (or whatever) is better tested and less sensitive than your servers.

# A recent disaster

- `gethostbyname` (“0124023429832498573298”) could, for a suitably large string, provide arbitrary code execution.
- Long-running (2001–15, at least) bug in GNU libc, with some pointer arithmetic not being done correctly
- You’ve got a problem. You use pointer arithmetic. Now you’ve got two problems.

# Use as remote attack

- SMTP mail protocol starts with the sender introducing themselves with “HELO my.host.name”.
- Protocol pre-dates DNS and this was only way to pass the name; in 1980 everyone was trusted to tell the truth.
- my.host.name is not normally checked (we just do a DNS reverse lookup on the source IP number these days) but some SMTP servers do use it as part of spam checking.
- Remote attack involves sending:
  - HELO 0124023429832498573298{and longer}\r\n
  - Buffer overrun / stack smash / etc

# DPI stops this

- DPI firewalls boxes look at SMTP sessions and perform some basic sanity checking, and only pass “sensible” instructions to the mail server.
- Prevent attacks involving weird character sets, buffer overruns, etc.
  - Cisco ASA limits all SMTP commands to 80 characters, while this attack requires at least 4096.
- However, in 2018, an awful lot of email is server-to-server encrypted (SMTP+TLS) so these defences don’t work. Decrypting the session is a risk/return decision.

# IPS/IDS

- Scans all packets on the network for problems
  - Mostly ancient exploits with static patterns, but on networks with poor patching they are still serious.
- Might maintain state about connections too
- Looks for trouble and shuts it down by either dropping packets (“in line”) or by instructing firewalls to add dynamic rules
- In many cases, false positive rate renders it unusable.
- Again, it would be interesting to see how many deployments are being used seriously.

# Network Security 7: Firewalls

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# What is a firewall?

- A device which looks at packet **headers** and blocks or passes them based on some policy.
  - Not based on packet contents
  - Attacker controls the entire packet
    - You cannot assume that the header is only constructed by the good guys

# Where do firewalls live?

- On routers
- On dedicated “firewalls” which are routers with a different/bigger software load
- On computers acting as routers
- On computers looking after themselves
- On computers acting as dedicated firewalls
  - Distinction between router and computer is generality of operating system and application mix

# What can firewalls do?

- Keep unwanted packets away from operating systems and applications
- Limit (imperfectly) which machines can send packets
- Keep unusual packets away from computers

# What can firewalls not do?

- Detect Malware / Trojans / Viruses / Etc
- Detect mis-use of legitimate protocols
- Stop attackers from crafting packets themselves
- Detect the malice or otherwise of packets
- Firewalls are not magic: “I have got a firewall” is not a protection against evil.

# What can a firewall do which an application or operating system cannot?

- **Nothing**
- This is very important: a firewall can only look at packets and take decisions, a job which every operating system does as those packets arrive

# So why do we have firewalls?

- Set policy at borders
  - Take some decisions out of computers' hands
  - Use hardened operating systems with smaller attack surfaces
  - Use simpler code than the full IP stack to “ride shotgun”
  - Rate limiting to protect under-powered computers
  - Pay homage to the concerns, and standards, of the 1990s (PCI-DSS, in particular)

# Operation of a firewall

- Receive a packet
- Look at the headers (particularly the source IP, the destination IP and the destination port)
- Optionally update some state (“stateful”)
- Pass or reject the packet

# TCP State: Revision

- A connection is proposed by the client sending a SYN with a proposed sequence number for the client's data.
- The server sends a packet with a SYN, a proposed sequence number for the server's data, and an acknowledgement of the client's SYN.
  - Alternatively, an RST to refuse the connection
- The client sends an ACK.
- SYN – SYN/ACK – SYN: the “three way handshake”

# TCP State: Revision

- Once we are communicating data, each side sends data with a sequence number, and we expect acknowledgements which are up to and including the last byte.
  - We send 1500 bytes starting at sequence number 20000, we expect acknowledgments up to 21500, but not later)
  - We expect acks to be monotonically increasing

# TCP State: Revision

- When a connection is finished with, FIN is sent.
- The other side ACK's the FIN, and passes an indication to the application.
- After possibly sending more data, a FIN is sent.
- And finally ACK'd.
- FIN – ACK – FIN – ACK, four packets to close a connection

# Connection State

- If we have seen a SYN go out, we expect a packet with a SYN and an ACK, or a packet with an RST, to come back, and nothing else.
- Once data is being transferred, we expect data and ACKs to be in rough step with each other
- We can similarly check the progress of FIN

# Typical implementations

- Solaris / OSX <10.8 / FreeBSD / etc “ipf”
- OpenBSD / OSX >=10.8 / Solaris >= 11 “pf”
- Extra code in either the ethernet interface or just behind it, which passes TCP packets to a set of state machines

# Sample Fragment

```
block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
        port = 25 flags S/SA head 101
pass in quick from 147.188.128.54/32 to any keep state group 101
pass in quick from 147.188.128.127/32 to any keep state group 101
pass in quick from 147.188.128.129/32 to any keep state group 101
pass in quick from 147.188.128.219/32 to any keep state group 101
pass in quick from 147.188.128.221/32 to any keep state group 101
pass in quick from 147.188.192.250/32 to any keep state group 101
pass in quick from 147.188.192.249/32 to any keep state group 101
```

# Sample Fragment

```
block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    flags S/SA head 102
# 5877 is obsolete clone of 587,
# 993 is obsolete imaps prior to use of STARTTLS verb
# block return-rst in quick from any to any port = 993 keep state group 102
block return-rst in quick from any to any port = 5877 keep state group 102
pass in quick from any to any port = 587 keep state group 102
pass in quick from any to any port = 993 keep state group 102
pass in quick from any to any port = 53 keep state group 102
pass in quick from any to any port = 143 keep state group 102
pass in quick from any to any port = 80 keep state group 102
pass in quick from any to any port = 443 keep state group 102
pass in quick from any to any port = 22 keep state group 102
pass in quick from 128.204.195.144 to any port = 2010 keep state group 102
pass in quick from 128.204.195.144 to any port = 2007 keep state group 102
pass in quick from 128.204.195.144 to any port = 2009 keep state group 102
pass in quick from 128.204.195.144 to any port = 2003 keep state group 102
```

# State is complex

src/dst

seq #

```
client-8-41.eduroam.lxuni.org.uk -> mail.barten.eu.org pass 0x40008502 pr 6 state 4/6
tag 0 ttl 11997
49341 -> 143 2b8fa4fa:adc9c1:65152<<5:63336<<1
cmsk 0000 smsk 0000 s0 2b8f6540/00ad83d2
FWD:ISN inc 0 sumd 0
REV:ISN inc 0 sumd 0
forward: pkts in 147 bytes in 23970 pkts out 0 bytes out 0
backward: pkts in 0 bytes in 0 pkts out 108 bytes out 24279
pass in quick keep state          IPv4
pkt_flags & 0(10000) = 1000,          pkt_options & ffffffff = 0, ffffffff = 0
pkt_security & ffff = 0, pkt_auth & ffff = 0
is_flx 0x1 0 0 0x1
interfaces: in @[mailprod0],@[] out @[],@[mailprod0]
```

# Value of this?

- In principle, implementations should discard out-of-state packets anyway
- Concern is Denial of Service and untested code-paths

# Denial of Service

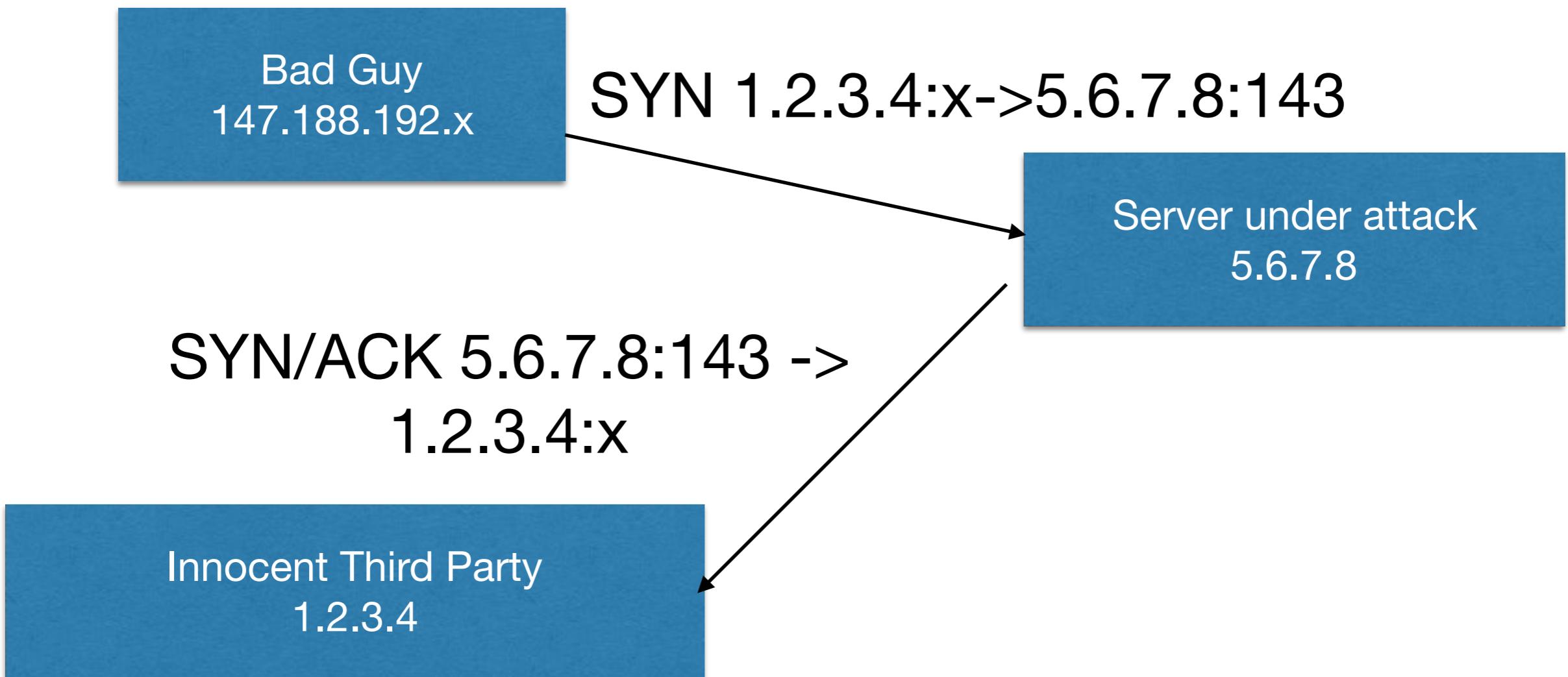
- Send a host a packet proposing a new connection to the IMAP server
  - 1.2.3.4:12345 -> 5.6.7.8:143, SYN 43279
- Assuming there is an IMAP server running, the server software is not informed of this until the three-way handshake is finished
- OS however acknowledges the SYN (SYN/ACK), generates its own sequence number, sets up a protocol control block and awaits the expected ACK from the client.
- “Client” is an attacker, forging packets.

# Impact #1

- The server that is receiving the requests runs out of kernel memory, because it keeps on setting up control blocks which are never used.
- Might take minutes for them to be reclaimed (spec allows 2 minutes for connection to set up)
- Attacker can set up a lot of control blocks in two minutes (potentially hundreds per second).

# Impact #2

- An innocent third party might get a high rate of SYN-ACKs from the server under attack and be unable to determine real source



# Firewalls can stop this

- Connection tracking allows it to count number of outstanding connection requests that have not been acknowledged
- Can set absolute thresholds, rates, etc
- Could be retrofitted to OS kernels, but would be a risk: sometimes this is (briefly) reasonable behaviour on a LAN

# Dealing with SYN floods

- Still a common DoS by unsophisticated attackers

# The problem

- Attacker constructs a packet with a SYN for a common port that will be accepted by firewalls, and a fake source address
- Server allocates a PCB and buffers on arrival of each SYN, and runs out of memory.
- SYN/ACK floods innocent third party: two for the price of one
- Fake source address can be a trusted source (so attack port 25 with fake MessageLabs address).
- No need to control source address as attacker does not need (indeed, does not want) the responses

# Firewall Solutions

- State tracking to limit the maximum number of incomplete connections per host
- Rate limiting to limit the number of new connections (probably a good idea anyway)

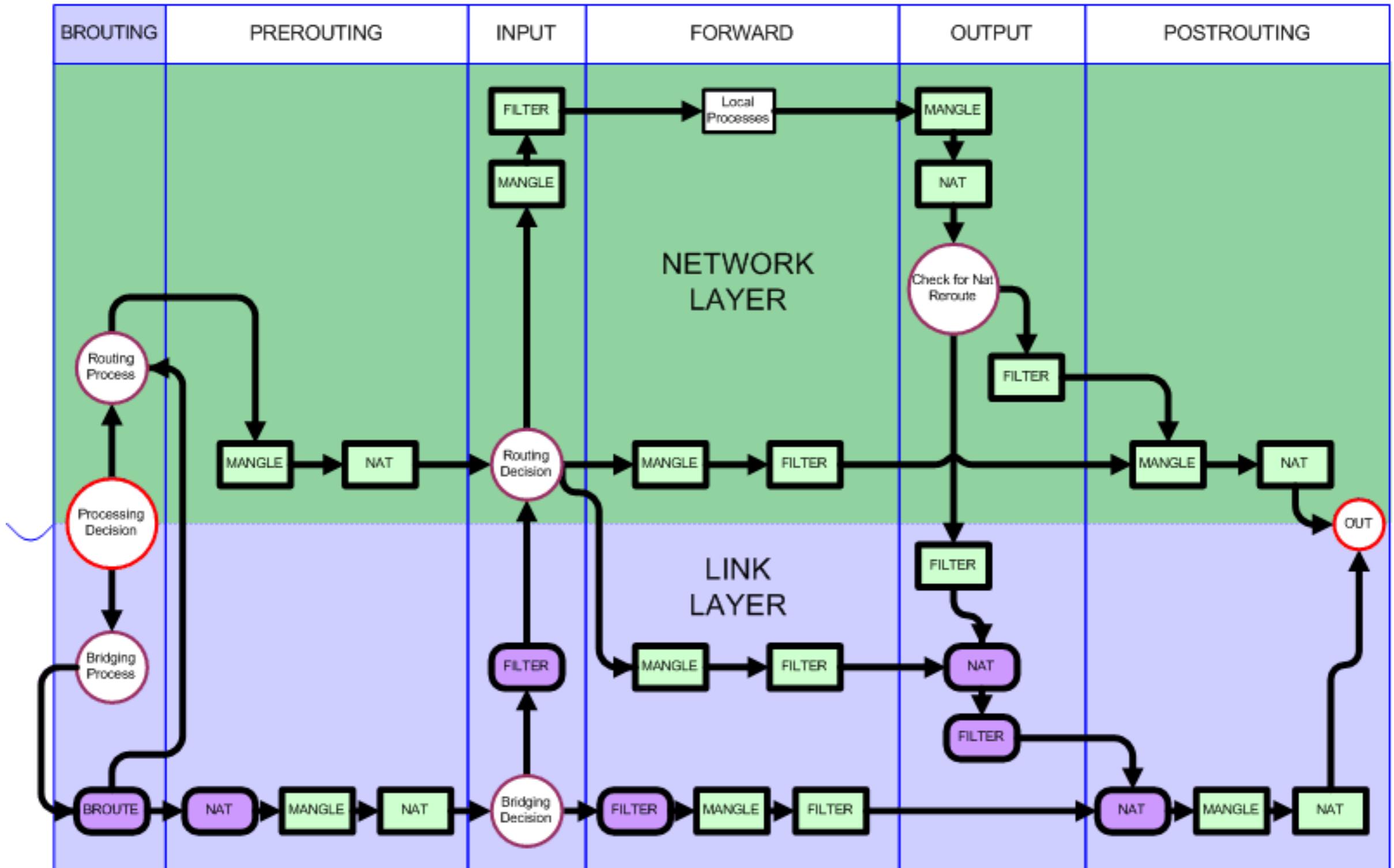
# Host solutions

- Timeout the PCBs more quickly
  - Breaks standards
- Defer allocation of full PCB until receipt of ACK of the SYN/ACK
  - Complex, invasive changes to kernel
  - Will introduce delay into every connection

# Firewalls on Linux

- Linux Firewalls are much more complex, as they sit in the IP stack, not in the ethernet drivers.
- The firewall code is called at a variety of points during the progress of a packet through the kernel

# I say complex...



# Slightly Simpler

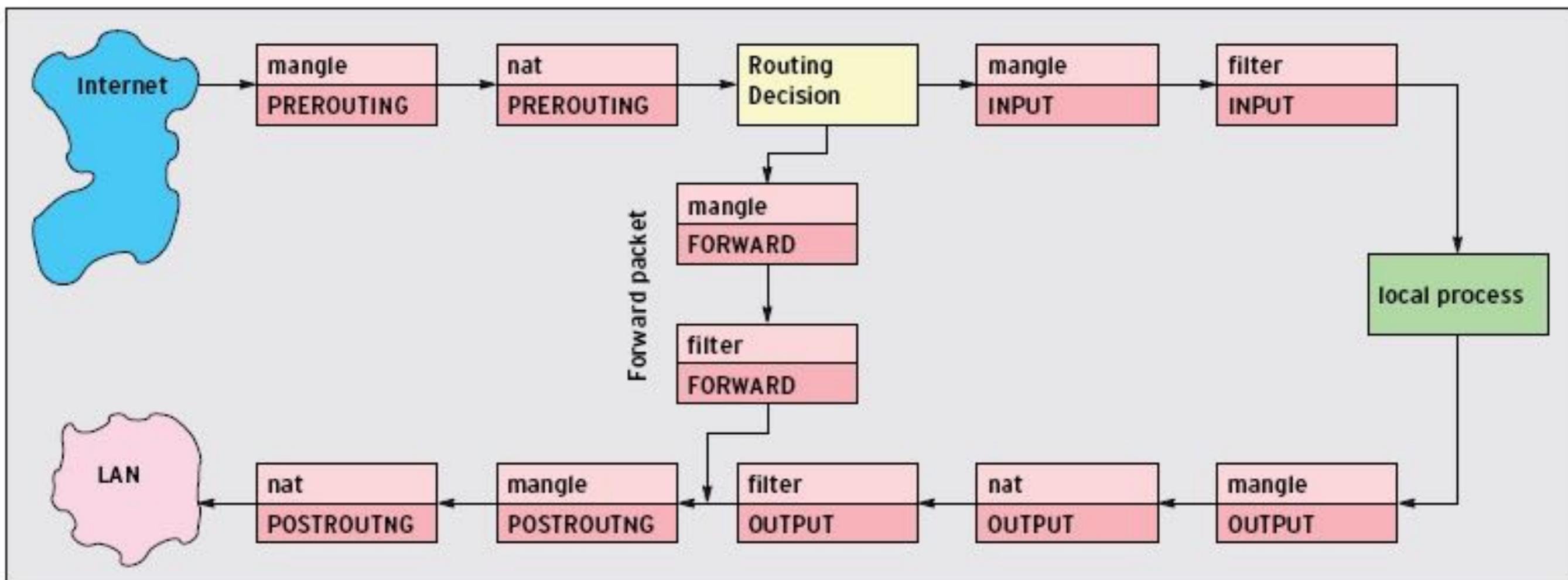
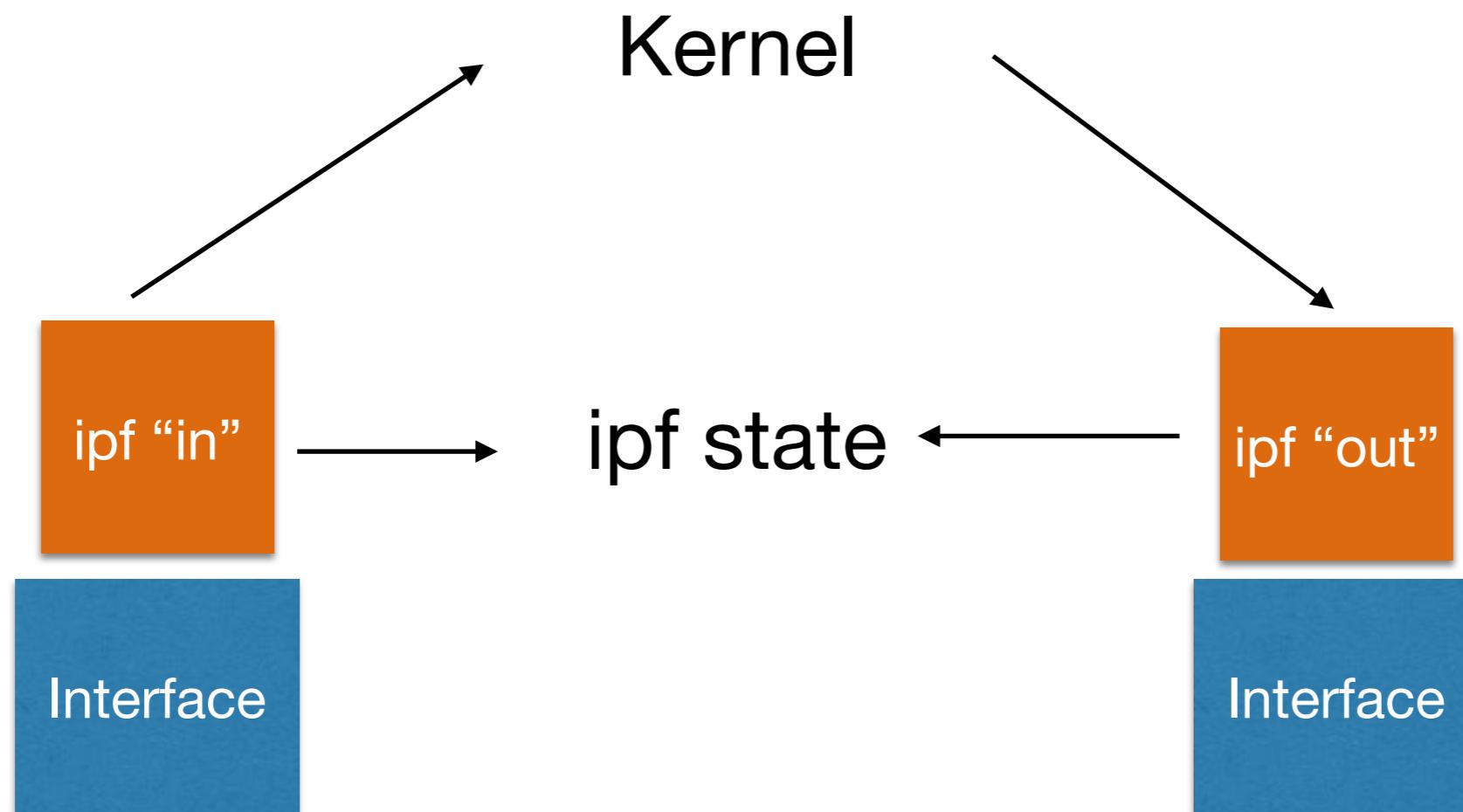


Figure 3: With Iptables, incoming packages first pass through the mangle table in the PREROUTING chain; the NAT table then follows. If the packet is destined for another host, the packet is simply passed to the mangle table in the POSTROUTING chain.

# Contrast with...



# Linux iptables

- Appears by name in Linux machines
- Architecture is borrowed, usually with implementation, for cheap routers that run Linux
  - For example, Mikrotik
- Sometimes with an added dash of ipchains, the earlier Linux implementation

# Linux iptables

- Mixture of:
  - “NAT” rulesets, which map addresses between internal and external (or whatever)
  - “mangle” rulesets, which can modify packets in a variety of (unexpected) ways and add marks for later use
  - “filter” rulesets, which can drop or accept packets

# Linux iptables

- Called
  - When packets arrive (“pre routing”)
  - When packets leave (“post routing”)
  - When packets are sent up to applications (“input”)
  - When packets come down from applications (“output”)
  - When packets are pass through by the machine acting as a router (“forward”)
- A given packet will go through exactly one of input, output or forward

# So...

- We can mangle, and then filter, packets in many different places
- And the packets will be in varying stages of processing at each point.
- Effective architectures use **marks**

# Input v Forward

- The split between “input” and “forward” can be confusing.
- By the time either is called, the decision has been taken whether to pass to application or send out of another interface.
- For a machine doing routing, the “input” policy on the “outside” network should probably be “block everything”. Remote management of routers is unwise.

# Practical Examples

- Simple enterprise firewall
- Policy is basically “drop everything we don’t explicitly need”
- Accepts various VPN protocols, ssh, and logging from places it expects logging from.
- Everything else goes on the floor

# Pre-Routing Mangle

```
[igb@rb2011-1] /ip firewall mangle> /ip firewall mangle print  
where chain=prerouting  
Flags: X - disabled, I - invalid, D - dynamic  
0    chain=prerouting action=jump jump-target=typecheck  
protocol=!tcp in-interface=pppoe-aa-uplink log=no log-  
prefix=""  
  
1    ;; Select on WAN interface and don't bother running  
these tests on other packets  
      chain=prerouting action=jump jump-target=ppoa-groom in-  
interface=pppoe-aa-uplink log=no log-prefix=""  
[igb@rb2011-1] /ip firewall mangle>
```

# Checking IP Types

```
[igb@rb2011-1] /ip firewall mangle> /ip firewall mangle print where  
chain=typecheck  
Flags: X - disabled, I - invalid, D - dynamic  
0 X chain=typecheck action=return protocol=tcp log=no log-prefix=""  
1 chain=typecheck action=return protocol=udp log=no log-prefix=""  
2 chain=typecheck action=return protocol=icmp log=no log-prefix=""  
3 chain=typecheck action=return protocol=ipsec-ah log=no log-  
prefix=""  
4 chain=typecheck action=return protocol=ipsec-esp log=no log-  
prefix=""  
5 chain=typecheck action=log log=no log-prefix="bad protocol"  
6 chain=typecheck action=mark-packet new-packet-mark=bogus  
passthrough=yes log=no log-prefix=""  
[igb@rb2011-1] /ip firewall mangle>
```

# Checking and Limiting

```
[igb@rb2011-1] /ip firewall mangle> /ip firewall mangle print where chain=ppoa-groom
Flags: X - disabled, I - invalid, D - dynamic
0    ;;; Mark bogus (RFC1918 etc) source input packets as bogus
      chain=ppoa-groom action=mark-packet new-packet-mark=bogus passthrough=yes src-address-list=bogons
log=no log-prefix=""
1    ;;; Bless AH
      chain=ppoa-groom action=mark-packet new-packet-mark=blessed passthrough=no protocol=ipsec-ah log=no
log-prefix=""
2    ;;; Bless ESP
      chain=ppoa-groom action=mark-packet new-packet-mark=blessed passthrough=no protocol=ipsec-esp log=no
log-prefix=""
3    ;;; TCP PSD
      chain=ppoa-groom action=mark-packet new-packet-mark=bogus passthrough=no protocol=tcp psd=21,3s,3,1
log=no log-prefix=""
4    ;;; UDP PSD
      chain=ppoa-groom action=mark-packet new-packet-mark=bogus passthrough=no protocol=udp psd=21,3s,3,1
log=no log-prefix=""
5    ;;; Mark all packets in invalid states as bogus
      chain=ppoa-groom action=mark-packet new-packet-mark=bogus passthrough=no connection-state=invalid
log=no log-prefix=""
6    ;;; Drop loose source routing
      chain=ppoa-groom action=mark-packet new-packet-mark=bogus passthrough=no ipv4-options=loose-source-
routing log=no log-prefix=""
7    ;;; Drop strict source routing
      chain=ppoa-groom action=mark-packet new-packet-mark=bogus passthrough=no ipv4-options=strict-source-
routing log=no log-prefix=""
[igb@rb2011-1] /ip firewall mangle>
```

# Input Ruleset

```
0    ;;; Accept everything for input chain that has not come from outside
     chain=input action=accept in-interface=!pppoe-aa-uplink log=no log-prefix=""

1    ;;; Drop packets with the bogus tag from pre-routing chain
     chain=input action=drop packet-mark=bogus log=yes log-prefix="bogus on input:"

2    ;;; Accept blessed packets
     chain=input action=accept packet-mark=blessed log=no log-prefix=""

3    ;;; Rate limit ICMP
     chain=input action=accept protocol=icmp in-interface=pppoe-aa-uplink dst-limit=1,5,dst-address/
1m40s log=no log-prefix=""

4    ;;; Accept established connections
     chain=input action=accept connection-state=established in-interface=pppoe-aa-uplink log=no log-
prefix=""

5    ;;; Accept protocols we use, with appropriate limits (rate of creation, total connections)
     chain=input action=accept connection-state=new protocol=tcp in-interface=pppoe-aa-uplink dst-
port=443,1194 limit=1,5 log=no
     log-prefix=""

6    ;;; Accept protocols that we use, with rate limit on packets from unknown sources
     chain=input action=accept connection-state=new protocol=udp in-interface=pppoe-aa-uplink dst-
port=1194,500,5678,1701,4500 log=no
     log-prefix=""

7    ;;; And drop everything else
     chain=input action=drop in-interface=pppoe-aa-uplink log=yes log-prefix="default input:"
```

# Forward Ruleset (1)

```
[igb@rb2011-1] /ip firewall mangle> /ip firewall filter print where chain=forward
Flags: X - disabled, I - invalid, D - dynamic
0    ;;; Accept everything that is not either in or out of the pppoa link
      chain=forward action=accept in-interface=!pppoe-aa-uplink out-interface=!pppoe-aa-uplink log=no
log-prefix=""

1    ;;; Forward everything outbound
      chain=forward action=accept out-interface=pppoe-aa-uplink log=no log-prefix=""

2    ;;; Drop everything in the ossec-list
      chain=forward action=drop src-address-list=ossec-list in-interface=pppoe-aa-uplink log=no log-
prefix=""

3    ;;; Forward all inbound established traffic
      chain=forward action=accept connection-state=established in-interface=pppoe-aa-uplink log=no log-
prefix=""

4    ;;; Drop packets with bogus mark from pre-routing
      chain=forward action=drop packet-mark=bogus log=yes log-prefix="bogus on forward:"

5    ;;; [ Weird and not interesting, deleted ]

6    ;;; Pass blessed packets from pre-routing
      chain=forward action=accept packet-mark=blessed log=no log-prefix=""
```

# Forward Ruleset (2)

```
7    ;;; Forward new https connections (mostly VPN) with rate limiting
     chain=forward action=accept connection-state=new protocol=tcp dst-address-list=https-
servers in-interface=pppoe-aa-uplink dst-port=443
     dst-limit=1,5,src-and-dst-addresses/1h log=no log-prefix=""

8    ;;; Permit ssh connections to appropriate places
     chain=forward action=accept connection-state=new protocol=tcp src-address-list=trusted-
sources dst-address-list=ssh-servers
     dst-port=22 dst-limit=3/1m,1,src-and-dst-addresses/1m40s log=no log-prefix=""

9    ;;; OSSEC to pi-two and ossec-sol
     chain=forward action=accept protocol=udp dst-address-list=ossec-sink in-interface=pppoe-
aa-uplink dst-port=1514 log=no log-prefix=""

10   ;;; tcp 514c for syslog to pi-two
     chain=forward action=accept connection-state=new protocol=tcp dst-address=81.187.150.213
src-address-list=offsite-locations
     in-interface=pppoe-aa-uplink dst-port=514 log=no log-prefix=""

11   chain=forward action=accept protocol=udp dst-address=81.187.150.213 src-address-
list=offsite-locations in-interface=pppoe-aa-uplink
     dst-port=514 log=no log-prefix=""

12   ;;; Drop inbound traffic that is out of state or otherwise unwanted
     chain=forward action=drop in-interface=pppoe-aa-uplink log=yes log-prefix="out of state on
forward"
```

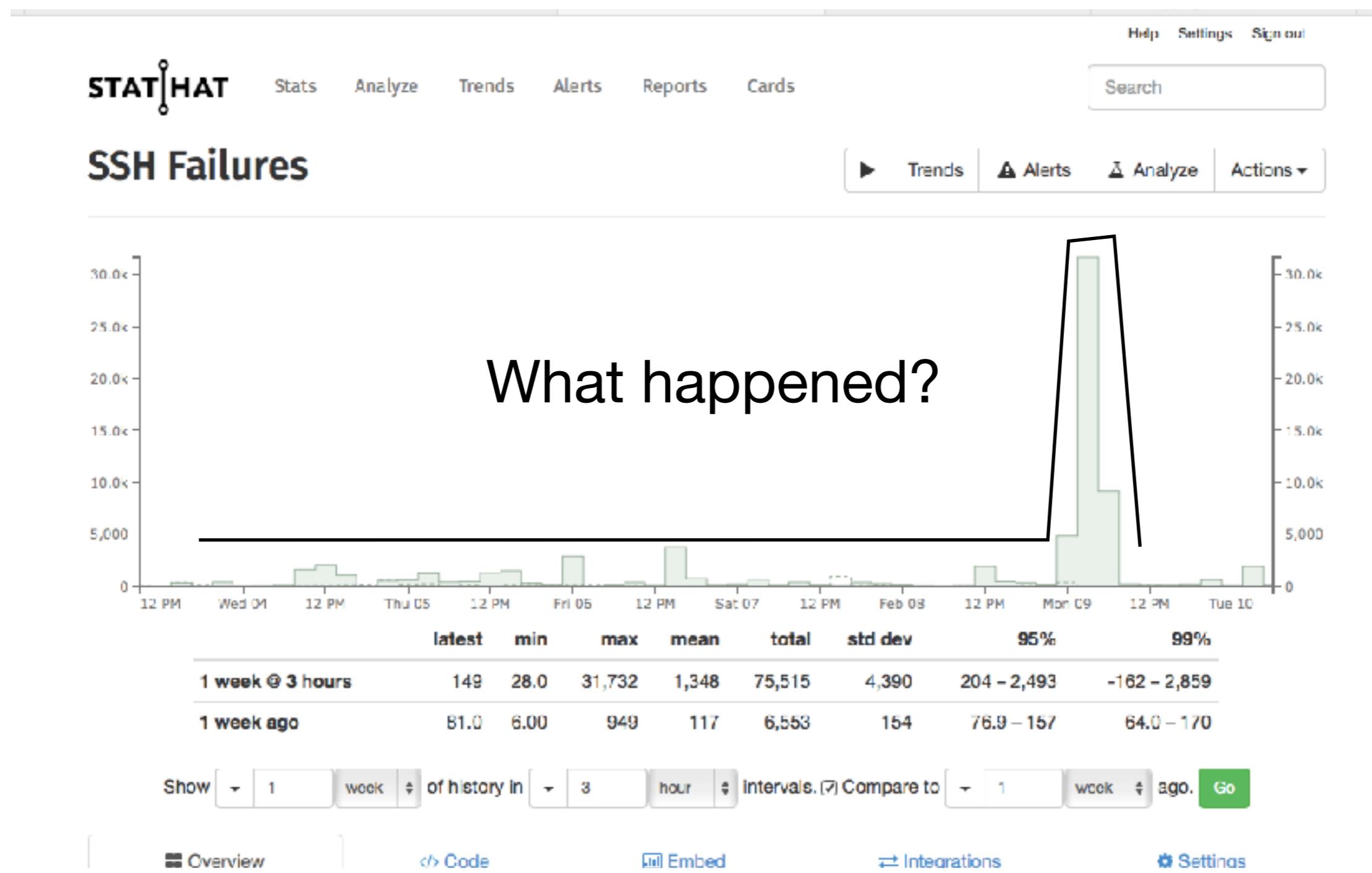
```
[igb@rb2011-1] /ip firewall mangle>
```



# Network Security 8: Virtual Firewalling

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Before we start...





[View my IP information](#)[More info about IPs](#)

Language ▾

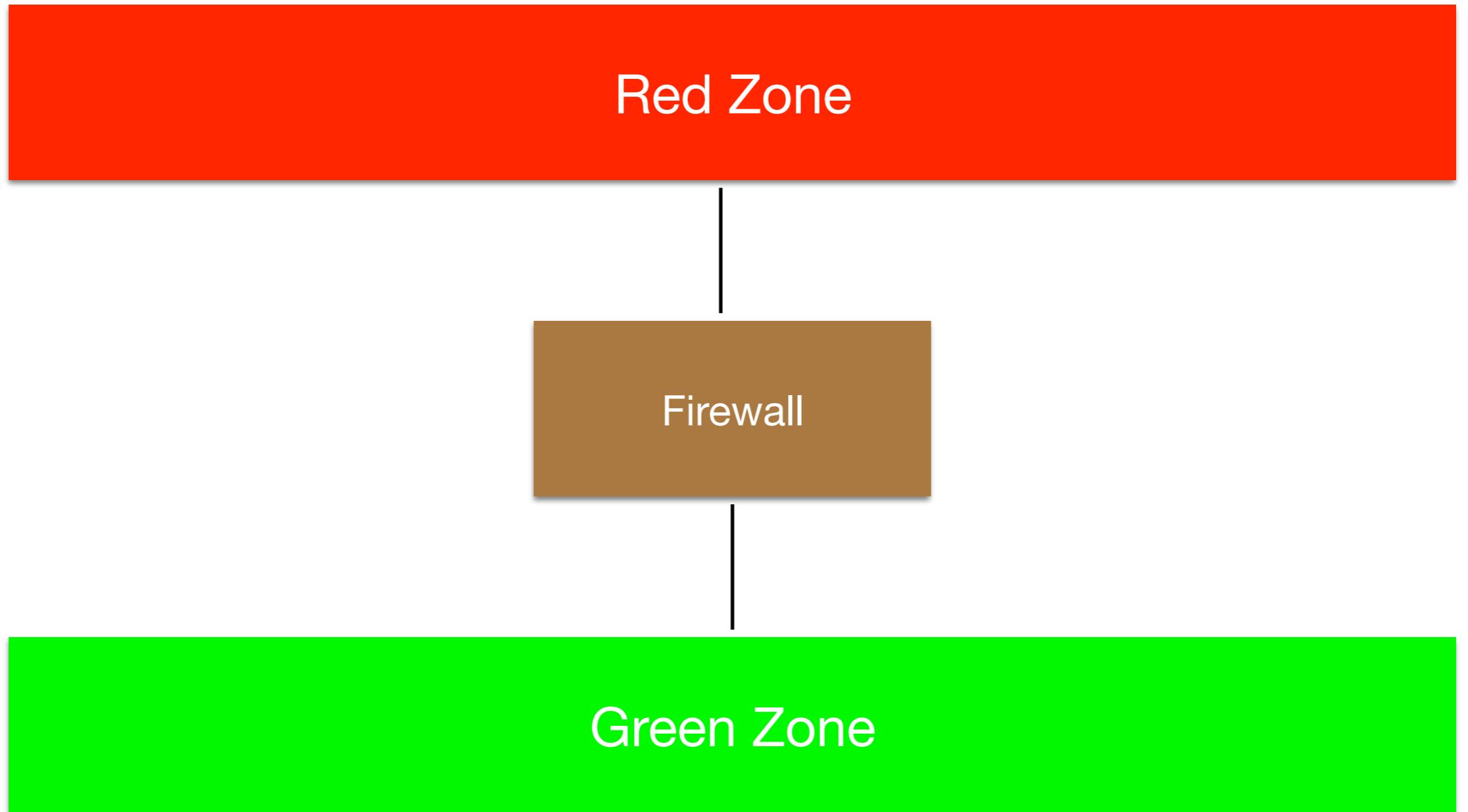
**Host/IP****Hostname:** 222.186.51.150**IP Address:** 222.186.51.150**Country:** China**Country Code:** CN (CHN)**Region:** Jiangsu**City:** Nanjing**Local time:** 10 Feb 22:47 (CST+0800)**Latitude:** 32.0617**Longitude:** 118.7778

Map data ©2015 Basarsoft, Google, ORION-ME, SK planet, ZENRIN

500 km

[Terms of Use](#)

# Physical Firewalls, Physical Networks



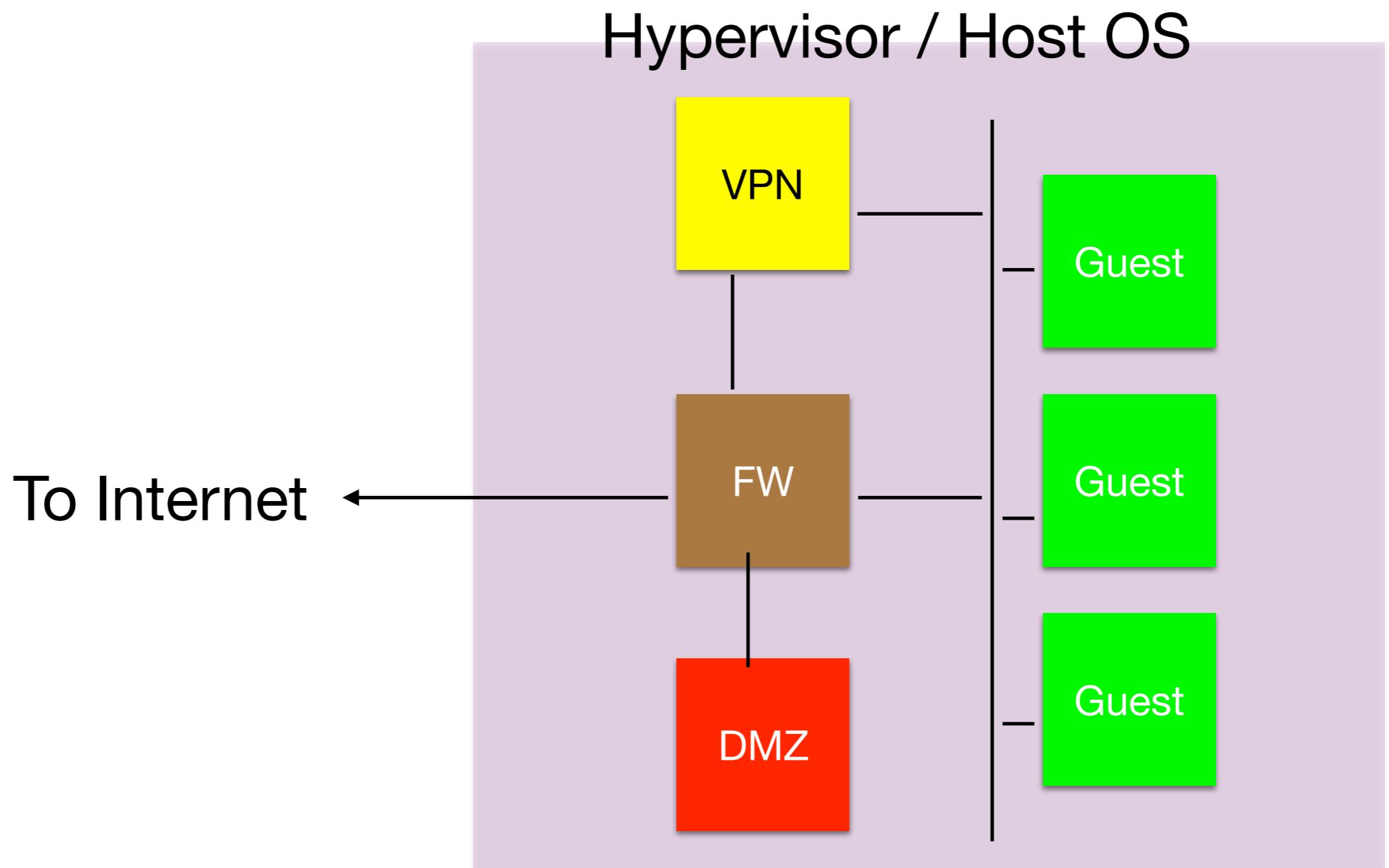
# Reality is Virtual

- **Virtual Networks**
- **Virtual Firewalls**
- **Virtual Hosts**
- All of the above can exist within one computer

# Purpose of Today

- Two different meanings of, and approaches to, virtualisation

# Virtualisation



# Definitions

- Problem with word “Virtual” is that it means many different things.
- In networking, a Virtual Network is a set of tags on a physical network
- In virtualisation, a Virtual Network is a purely software construct

# VLANS

- An extra “tag” inserted into the Ethernet packet format, saying which network the packet belongs to.
  - 4 extra bytes ahead of type/size fields, first 16 bits 0x8100 to unambiguously mark “this is a tag” (no real untagged packet will have 0x8100 there), 3 bits of priority, 1 bit to specify if frame is droppable, 12 bits for tag.
  - Tag 0 is equivalent to untagged, tag 1 is often used internally by switches, tag 0xFFFF (4095) is reserved.
- In principle, MAC addresses only need to be unique on a per-tag basis, but relying on this will break lots of switches.

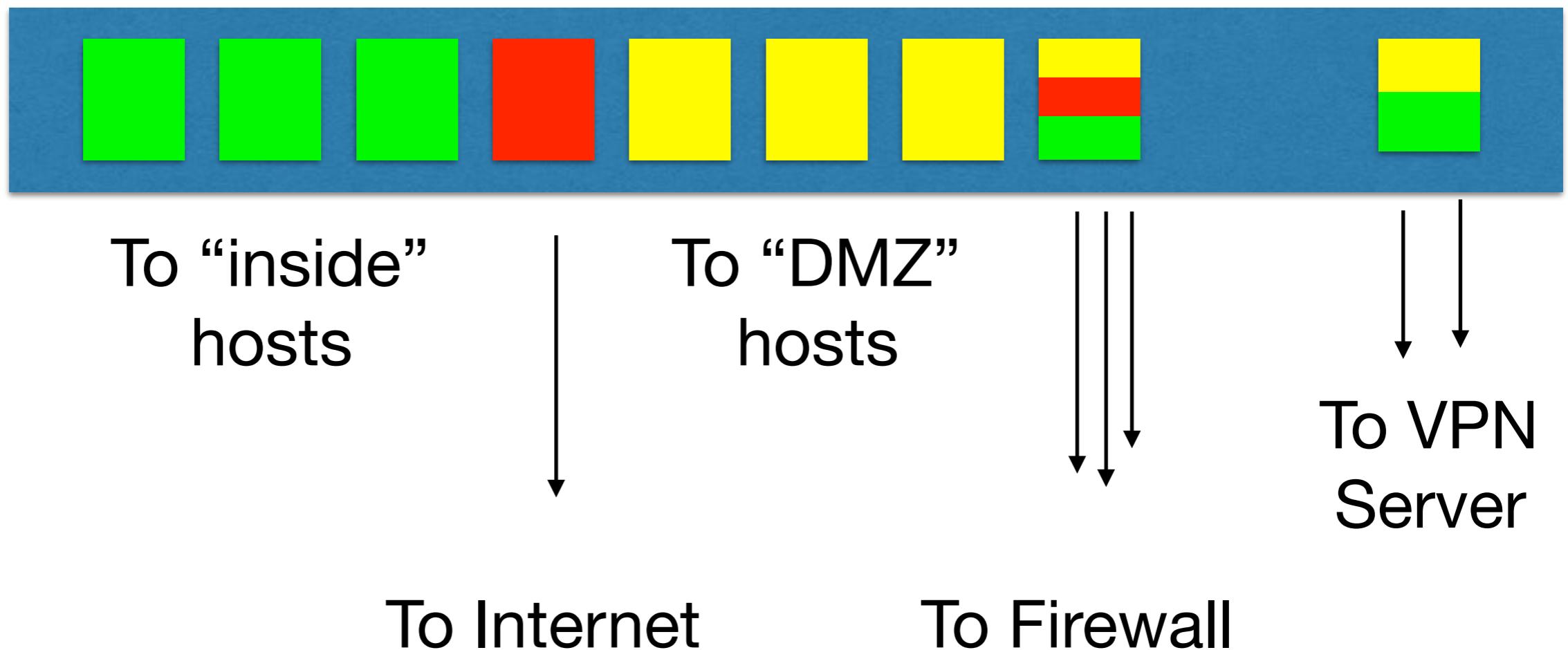
# VLANS allow...

- Trunking of multiple networks along one cable
- Trunking of multiple networks through one interface
- Segregation of traffic by type, security label, etc

# Common scenario

- Some ports of a switch “trunked” and carrying multiple VLANs with tagged packets, either to other switches, or VLAN-aware routers, firewalls, hosts.
- Some ports of a switch only carrying the default VLAN
- Some ports of a switch carrying a single non-default VLAN, untagged

# Switch Config



# Exercise Request

- I forgot to deal with the issue of interface names.
- If you can, it would be helpful if when you refer to interfaces in your script, you use variables, which you initialise at the top of your script with:
  - `$inside_or_whatever=${SERVERNET-eth0}`
  - `$outside_or_whatever=${CLIENTNET-eth1}`
- Where eth0 and eth1 are whatever your interfaces are called on your machine. That way, we can set the `$SERVERNET` and `$CLIENTNET` environment variables to make people's scripts work correctly, while your script continues to work normally.

# Exercise Request

```
#!/bin/bash
```

```
inside=${CLIENTNET}-eth0  
outside=${SERVERNET}-eth1
```

```
iptables -A FORWARD -i $inside ...  
iptables -A FORWARD -o $outside ...  
iptables -A FORWARD -i !$inside ...
```

```
# and so on
```

# Shell fun

- More generally, the things you can do (bash and ksh) with variables by writing \${variable□thing} where □ is punctuation are horrifyingly useful.

```
$ foo=XYZZY
$ echo $foo ${foo%Z*} ${foo%%Z*} ${foo:2:2} ${foo/X/A}
XYZZY XYZ XY ZZ AYZZY
$
```

# Port Types

- “Tagged” or “Trunk” ports: each packet is marked with its tag (in sane networks, the global tag for its network: you can change the tags on a per-port basis, but you should not do this).
- “Untagged” or “Access” ports: packets are untagged, even if they originate from a non-default network.

# Lots of ways to “see” VLANS

- You can get at them directly, as on (some) Linuxes:

Note MAC addresses

```
igb@pi-one:~$ ifconfig -a
eth0      Link encap:Ethernet HWaddr b8:27:eb:e1:96:51
            inet addr:10.92.213.231 Bcast:10.92.213.255 Mask:255.255.255.0
            inet6 addr: 2001:8b0:129f:a90f:ba27:ebff:fe00:efe7/64 Scope:Global
            inet6 addr: fe80::ba27:ebff:fee1:9651/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:88366 errors:0 dropped:0 overruns:0 frame:0
            TX packets:69956 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:8540091 (8.1 MiB) TX bytes:12480470 (11.9 MiB)
```

```
eth0.5    Link encap:Ethernet HWaddr b8:27:eb:e1:96:51
            inet addr:81.187.150.211 Bcast:81.187.150.223 Mask:255.255.255.240
            inet6 addr: 2001:8b0:129f:a90e:ba27:ebff:fe00:efe7/64 Scope:Global
            inet6 addr: fe80::ba27:ebff:fee1:9651/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:29632 errors:0 dropped:0 overruns:0 frame:0
            TX packets:27889 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:3565578 (3.4 MiB) TX bytes:5562868 (5.3 MiB)
```

“eth0” is  
untagged  
traffic

“eth0.5” is  
traffic using  
VLAN tag 5

# Virtual Interface per tag

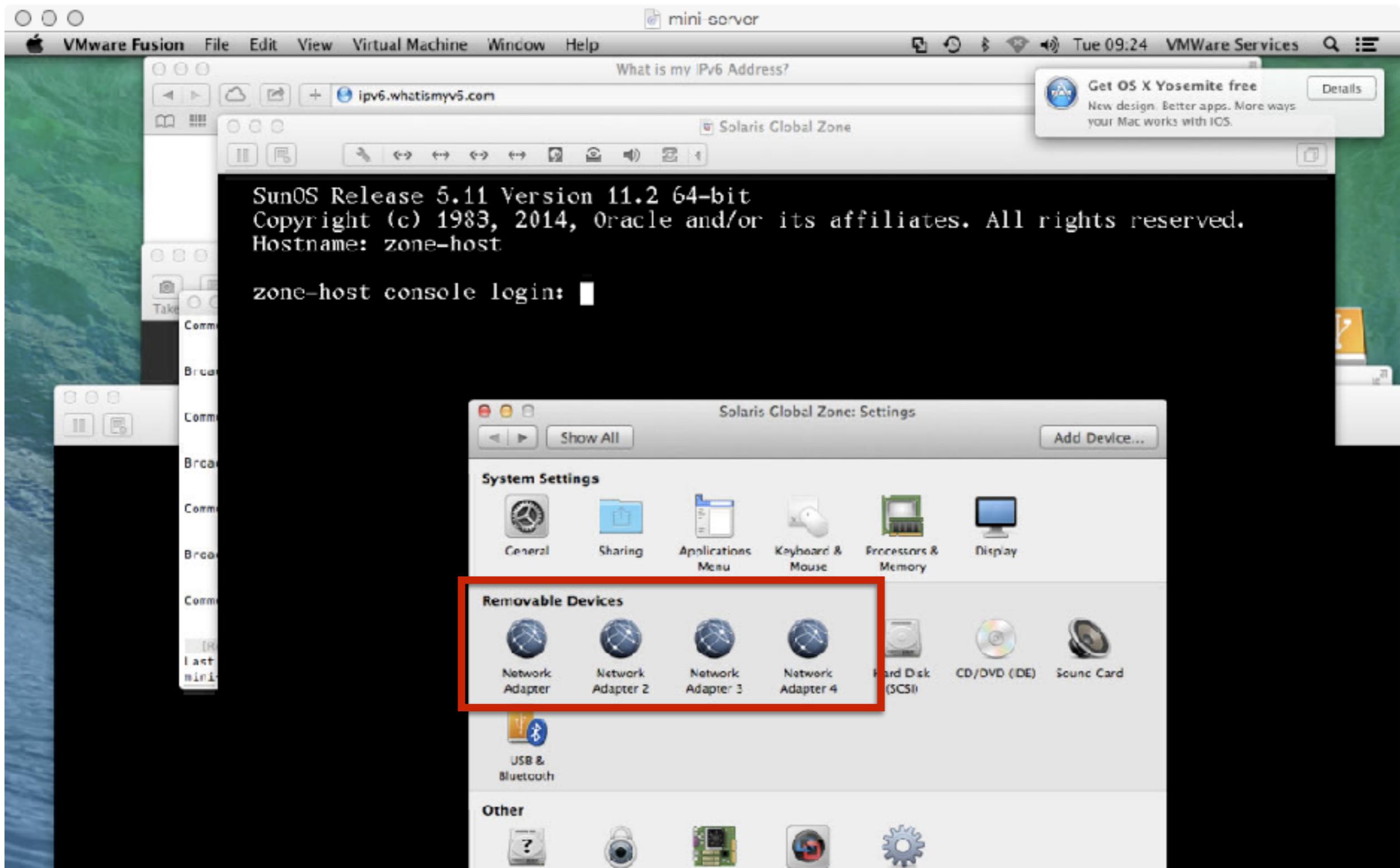
- You can see them as virtual interfaces, as on (modern) Solaris
- The physical link is the interface, then there are multiple virtual interfaces, one per tag

```
igb@research-1:~$ dladm  
LINK          CLASS      MTU      STATE      OVER  
net0          phys       1500     up        --  
vnic6         vnic       1500     up        net0
```

```
igb@research-1:~$ dladm show-vnic vnic6  
LINK          OVER      SPEED      MACADDRESS      MACADDRTYPE VIDS  
vnic6         net0      1000      2:8:20:89:b5:a0  random      4008
```

```
igb@research-1:~$ ifconfig -a  
net0: flags=100001000943<UP,BROADCAST,RUNNING,PROMISC,MULTICAST,IPv4,PHYSRUNNING> mtu 1500 index 2  
      inet 147.188.192.246 netmask ffffff00 broadcast 147.188.192.255  
vnic6: flags=120202000841<UP,RUNNING,MULTICAST,IPv6,CoS,PHYSRUNNING> mtu 1500 index 3  
      inet6 fe80::8:20ff:fe89:b5a0/10
```

# VLANs in Virtualisation



# Tags have all been stripped

```
igb@zone-host:~$ dladm
LINK          CLASS      MTU   STATE    OVER
net1          phys       1500  unknown  --
net0          phys       1500  up       --
net2          phys       1500  up       --
ossec/net2    phys       1500  up       --
net3          phys       1500  up       --
ossec/net3    phys       1500  up       --
ossec/net0    vnic      1500  up       net0
igb@zone-host:~$
```

# Good uses for VLANs

- Reducing the number of physical cables and interfaces used between a switch and a firewall
- Reducing the number of physical switches (you can use different tags with only access ports to split a single switch between disjoint networks, getting economies of scale)
- Bringing multiple networks into machines with insufficient physical interfaces (general case of firewall).

# VLANs for segregation of management

- Telecoms practice divides hardware into three “planes”. It’s not common as a distinction in IT, but it’s a useful abstraction.
  - Management
  - Control
  - Data

# Data Plane

- The actual switching of data, at speed and scale.  
Equivalent to the ethernet ports on an ethernet switch.

# Control Plane

- Setting up calls, determining routes, and other less frequent, potentially higher impact, but usually automatic tasks
- Doesn't always have a direct IT equivalent, but routing protocols like OSPF and BGP would fall into this category.
  - Not TCP SYN SYN/ACK ACK

# Management Plane

- Reconfiguration of devices by manual action or by action of higher-level management systems
- Has ability to reroute traffic, shut down or reconfigure interfaces, etc, etc.
- Real telecoms equipment does not allow “in-band management” – you cannot cross to the management plane from the data plane.

# Building a Management Plane

- Some very high-end, specialised equipment does have a separate management port, through which the equipment can be managed.
- It's rare for that port to be the **only** way to manage the device, and running separate cabling is a pain
- With care, you can use VLANs to get much of the benefit

# Management VLAN

- Only listen for management traffic on one particular VLAN: packets for management must have that tag
- Why doesn't this work without more care?

# VLANs are insecure

- Anyone can put any tag on any packet
- VITAL that you police tags where they enter “trusted” (roughly, physically secure) parts of your network, by stripping tags that are not expected from edge ports

# 802.1x might have a role

- There are various dynamic solutions which allow you to configure switch ports based on MAC addresses, authentication material and so on.
- They are messy and tricky to get right (see previous discussion on 802.1x) but the alternatives can be messy as well.
- Planning a VLAN infrastructure requires a lot of thought.
- If I had my time again, I would not use default VLAN, and would tag **everything** except on access ports.
  - Usually you aren't starting from a green field, and no-one uses VLAN tagging on their first switch.

# Network Security: Practical Work

## (Not used in 2016–17)

[i.g.batten@batten.eu.org](mailto:i.g.batten@batten.eu.org)

# Solaris NIC Setup

```
igb@solaris:~$ sudo dladm create-etherstub ether0  
Password:  
igb@solaris:~$ sudo dladm create-vnic -l ether0 test0  
igb@solaris:~$ sudo dladm create-vnic -l ether0 test1
```

LINK	CLASS	MTU	STATE	OVER
net0	phys	1500	up	--
ether0	etherstub	9000	unknown	--
test0	vnic	9000	up	ether0
test1	vnic	9000	up	ether0

Result: a stub ethernet, with two interfaces plugged into it.

# Solaris Zone Setup

```
igb@solaris:~$ sudo zonecfg -z secondzone
Use 'create' to begin configuring a new zone.
zonecfg:secondzone> create
create: Using system default template 'SYSdefault'
zonecfg:secondzone> add net
zonecfg:secondzone:net> set physical=test1
zonecfg:secondzone:net> end
zonecfg:secondzone> verify
zonecfg:secondzone> commit
zonecfg:secondzone>
igb@solaris:~$ sudo zoneadm -z secondzone install
The following ZFS file system(s) have been created:
    rpool/VARSHARE/zones/secondzone
Progress being logged to /var/log/zones/zoneadm.20150212T130046Z.secondzone.install
    Image: Preparing at /system/zones/secondzone/root.

Install Log: /system/volatile/install.8249/install_log
AI Manifest: /tmp/manifest.xml.aRa0fq
SC Profile: /usr/share/auto_install/sc_profiles/enable_sci.xml
Zonename: secondzone
Installation: Starting ...

        Creating IPS image
Startup linked: 1/1 done
```

# Zone Setup

- Will now run for some time dragging over the packages needed to install the zone
- They're cached, so if you build more zones it'll be a lot quicker

# Zone Finishing

Updating package cache	0/0
Updating image state	Done
Creating fast lookup database	Done
Updating package cache	1/1
Installation: Succeeded	

Note: Man pages can be obtained by installing `pkg:/system/manual` done.

Done: Installation completed in **852.406** seconds.

Next Steps: Boot the zone, then log into the zone console (`zlogin -C`) to complete the configuration process.

Log saved in non-global zone as `/system/zones/secondzone/root/var/log/zones/zoneadm.20150212T130046Z.secondzone.install`  
igb@solaris:~\$

# Access the Zone

```
igb@solaris:~$ sudo zoneadm -z secondzone boot  
Password: // because it's been more than five minutes  
igb@solaris:~$ sudo zlogin -C secondzone  
[Connected to zone 'secondzone' console]
```

Counts down setting some things up, and then drops you into a configuration screen

# Follow the obvious prompts

## System Configuration Summary

Review the settings below before continuing. Go back (F3) to make changes.

Computer name: secondzone

Network:

Manual Configuration: test1/v4  
IP Address: 192.168.141.100/24  
Netmask: 255.255.255.0

Time Zone: UTC

Locale:

Default Language: English  
Language Support: English (United States)  
Username: igb

Support configuration:

Not generating a Support profile as OCM and ASR services are not installed.

# What do you have?

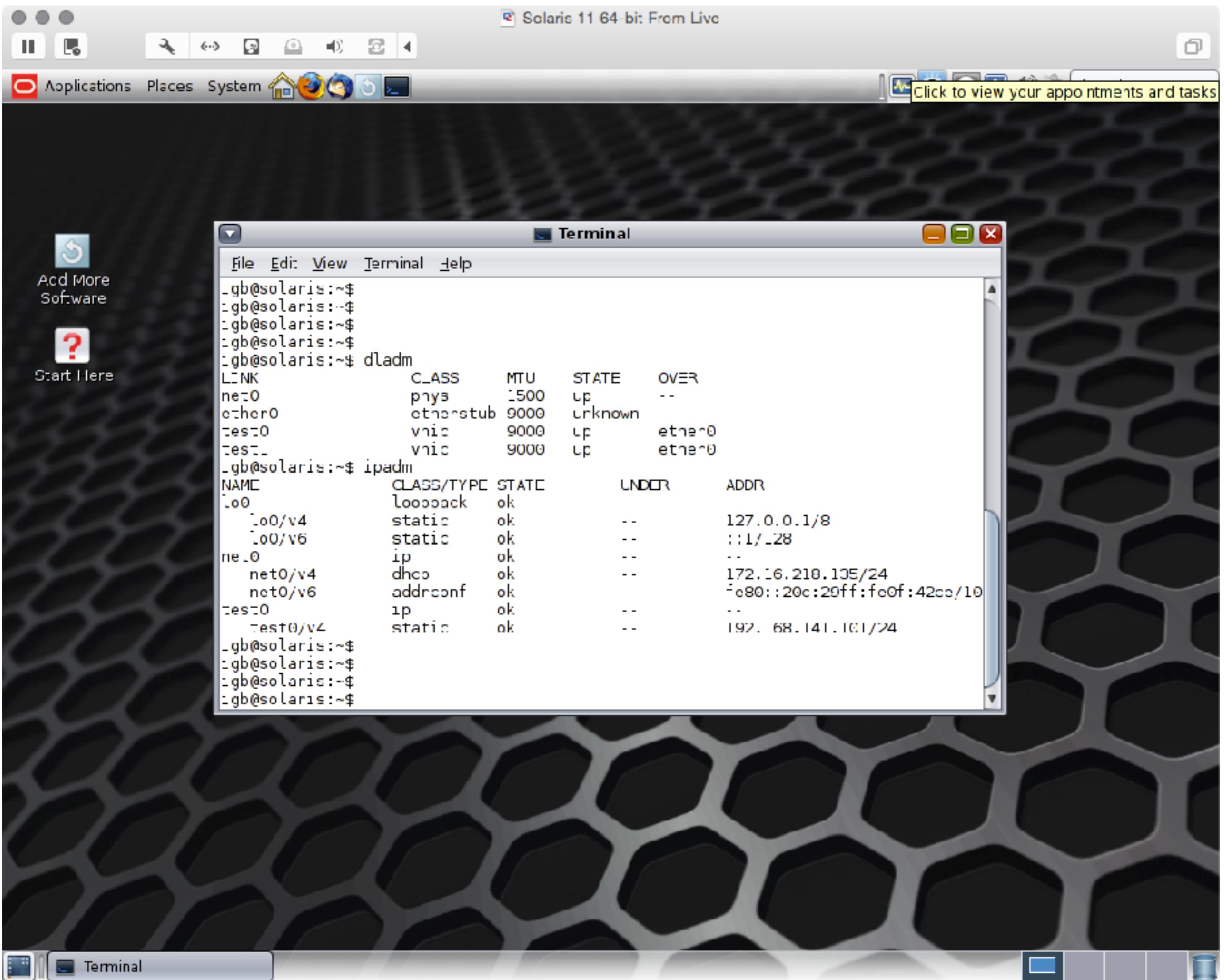
- secondzone is a running Solaris instance (let's not worry about the precise details of what “instance” means here)
- IP address 192.168.141.100, purely internal to the Solaris VM

# Now set up the other interface

```
igb@solaris:~$ sudo ipadm create-ip test0
igb@solaris:~$ sudo ipadm create-addr -T static -a 192.168.141.101/24 test0
test0/v4
igb@solaris:~$ ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
net0: flags=100001004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,PHYSRUNNING> mtu 1500 index 5
    inet 172.16.218.135 netmask ffffff00 broadcast 172.16.218.255
test0: flags=100001000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,PHYSRUNNING> mtu 9000 index 6
    inet 192.168.141.101 netmask ffffff00 broadcast 192.168.141.255
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
    inet6 ::1/128
net0: flags=120002004841<UP,RUNNING,MULTICAST,DHCP,IPv6,PHYSRUNNING> mtu 1500 index 5
    inet6 fe80::20c:29ff:fe0f:42ce/10
test0: flags=120002000840<RUNNING,MULTICAST,IPv6,PHYSRUNNING> mtu 9000 index 6
    inet6 ::/0
igb@solaris:~$
```

# So it's a network!

```
igb@solaris:~$ ssh 192.168.141.100
The authenticity of host '192.168.141.100 (192.168.141.100)' can't be established.
RSA key fingerprint is de:32:e9:88:9e:2e:61:66:2e:24:a7:a7:4c:6c:c8:b7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.141.100' (RSA) to the list of known hosts.
Password:
Last login: Thu Feb 12 13:20:03 2015
Oracle Corporation      SunOS 5.11      11.2      June 2014
igb@secondzone:~$
```



# Exercises

- Read the manual pages for “ipf”, “ipfstat” and the configuration files (“man ipf”, “man ipfstat” and “man -s4 ipf”)
  - Not even Unix obsessives defend the manual page “section” nonsense
- You can then edit a firewall into /etc/ipf/ipf.conf
- Load it with ipf -Fa -f /etc/ipf/ipf.conf
- Flush it with ipf -Fa

# For example

- Set up a firewall on secondzone to block access to the ssh server
  - You might want to use “flags S/SA”

# Default Deny

```
block return-rst in log first level local1.info quick \
  on test0 proto tcp from any to any \
  flags S/SA head 102
pass in quick from any to any port = 22 keep state group 102
```

You can add more similar “pass” lines to permit more  
tcp protocols

How would you test that this is working correctly?

# Tools

- Install nmap into the global zone
  - “sudo pkg install nmap”
- Read the manual page for it, and use it to see which ports are open on secondzone (you’ll probably have to quote secondzone’s IP number directly, or you can edit /etc/hosts)

# Tools

- Also look back at previous lectures and see use of netstat to look for ports in state LISTEN, and read the manual page of pfiles to see how to look at individual processes

# Exercise

- Write a default-deny firewall for Solaris which permits:
  - outbound DNS and the responses
  - inbound ssh
  - outbound TCP connection
  - Default Deny

# Security 11: Proxies and similar

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Firewalls: Not Enough

- Firewalls restrict access to services (or, more accurately, to ports on which services run).
- Provide no checking of correct behaviour inside protocol.
- Provide no protection against tunnelling, variant ports, etc, etc.
- Can block access to external services, but not very well (I just run my VPN/ssh/etc server on port 80)

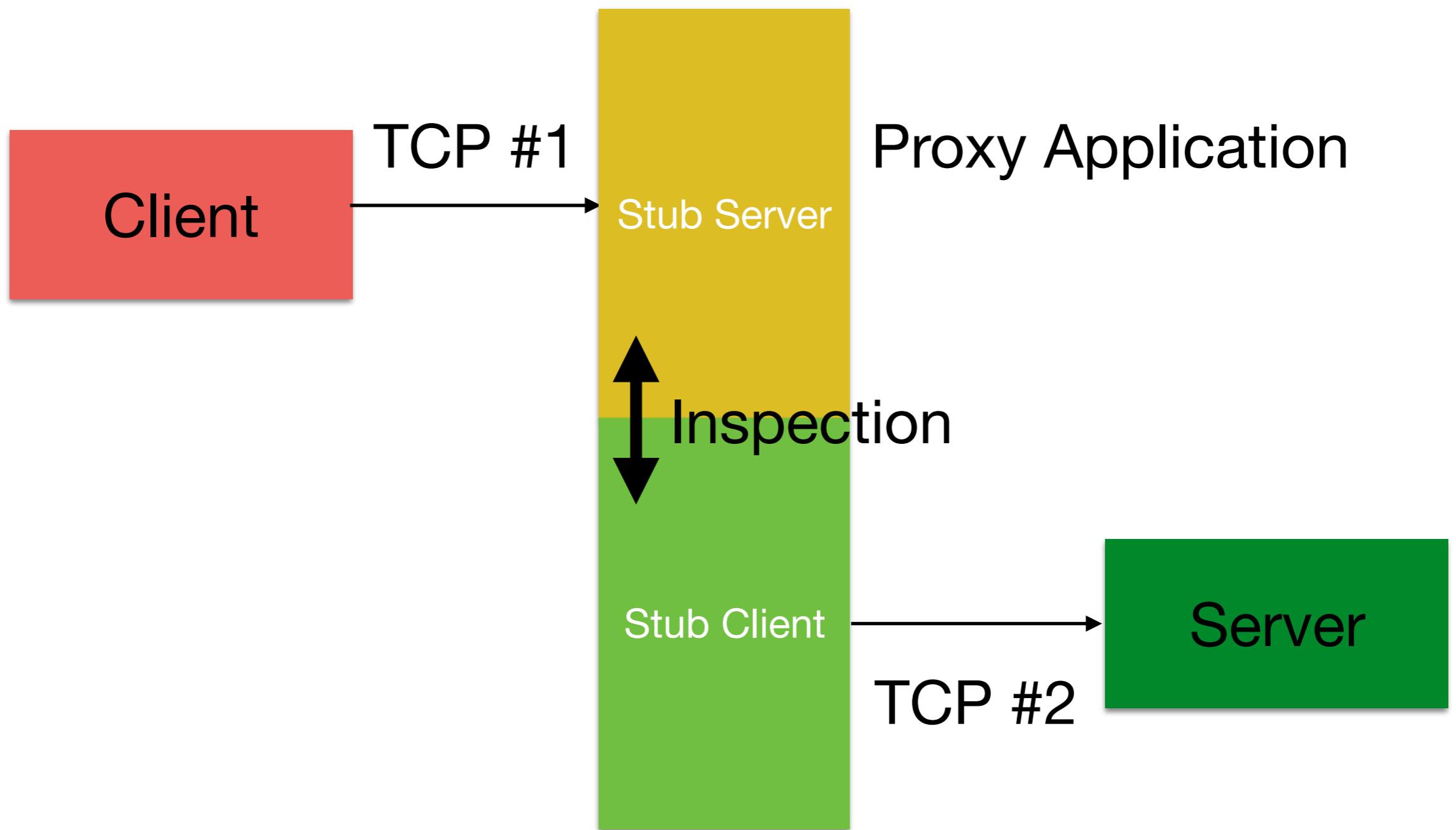
# NAT: Not Enough

- Provides no outbound security at all
- Provides some inbound security
- No checks on protocol operation

# Proxies

- Old technology: “fwtk” (confusingly, firewall toolkit) made available by TIS in 1993
- Extremely effective when used carefully

# Basic Premise



# Security Concept

- Stub server supports enough of the protocol to receive commands and check them for legality and plausibility (state, “usual size”, etc)
- Stub client supports enough of the protocol to send commands and receive responses
- Both are simple, clear, auditable code (you hope)
- Shields messy, attackable, complex, privileged origin server from attacker

# SMTP exchange

```
igb986@cs.bham.ac.uk... Connecting to smart1.bham.ac.uk. via relay...
220 smart1.bham.ac.uk ESMTP Exim 4.82 Tue, 17 Feb 2015 14:21:15 +0000
>>> EHLO mail.batten.eu.org
250-smart1.bham.ac.uk Hello mail.batten.eu.org [147.188.192.250]
250-SIZE 104857600
250-8BITMIME
250-PIPELINING
250 HELP
>>> MAIL From:<igb@batten.eu.org> SIZE=128
250 OK
>>> RCPT To:<igb986@cs.bham.ac.uk>
>>> DATA
250 Accepted
354 Enter message, ending with "." on a line by itself
>>> .
250 OK id=1YNj1X-0000U8-G7
igb986@cs.bham.ac.uk... Sent (OK id=1YNj1X-0000U8-G7)
Closing connection to smart1.bham.ac.uk.
```

# HTTP Exchange

```
wget -d -v http://www.batten.eu.org/~igb/fdsfasfs
```

```
Resolving www.batten.eu.org (www.batten.eu.org)... 2a00:7b80:3019:12::579c:4928, 128.204.195.144
Caching www.batten.eu.org => 2a00:7b80:3019:12::579c:4928 128.204.195.144
Connecting to www.batten.eu.org (www.batten.eu.org)|2a00:7b80:3019:12::579c:4928|:80... connected.
```

```
GET /~igb/fdsfasfs HTTP/1.1
User-Agent: Wget/1.14 (solaris2.11)
Accept: */*
Host: www.batten.eu.org
Connection: Keep-Alive
```

```
HTTP/1.1 404 Not Found
Server: nginx/1.0.15
Date: Tue, 17 Feb 2015 14:24:55 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
```

# Vectors...

- Mail server looks up supplied hostname (in some cases) and supplied email addresses (eventually)
- If those somehow trigger buffer overruns or similar (there was such an attack recently) then the receiver's SMTP daemon can be taken over by attacker
- Similarly HTTP: reverse lookups of addresses, lookups of URLs, calls to namei(), possible shell processing for cgi-bin, etc, etc, etc.

# In and Out

- Proxies can be used both inbound and outbound
- Inbound proxies cannot make assumptions about client knowing the proxy is there.
- Outbound proxies can use modified version of protocol, as clients can be configured to know.

# Example: Web proxying

- Outbound web proxying is used in place of NAT.
- Client connects to proxy, passes URL, proxy fetches content, sends content back.
- Makes tunnelling **much** harder: tunnelled data must look like plausible HTTP.
- Proxy can do content filtering, virus scanning, logging, etc, etc.

# Web Proxying

- 1995–2005, Web Proxying also offered **caching**.
- Repeated requests for same content served from cache to save bandwidth, improve performance.
- Benefits now very limited, most caching turned off by 2010.

# Web Proxying

- With configuration, proxy is known to client.
- For use without configuration, outbound “transparent proxying” involves NAT-ing packets destined for outside to proxy, and delivering (somehow) the intended destination so proxy can make onward connection.
  - Popular in mobile phone networks.

# Encryption

- Web proxies struggle with encryption.
- Solution #1: “CONNECT” operation allows client to make arbitrary connection through proxy which is relayed, byte-for-byte.
  - End to end encryption
  - Breaks security model: useful for avoiding NAT, but not for secure environments.

# Encryption

- Solution #2: Certificate Forgery
- Proxy performs man in the middle attack by creating certificate on the fly for requested site, so proxy has access to plaintext.
- Used for content filtering in schools, etc.

# Inbound Proxies

- For HTTP, proxy can check for syntactically correct protocol and scan for obvious attacks, passing only sanitised requests to internal servers.
- Can also do basic access control in one place, when “origin servers” cannot be trusted to do it right.
- Proxy can also scan content for sensitive material (“Data Leakage Protection”)

# Inbound Proxies

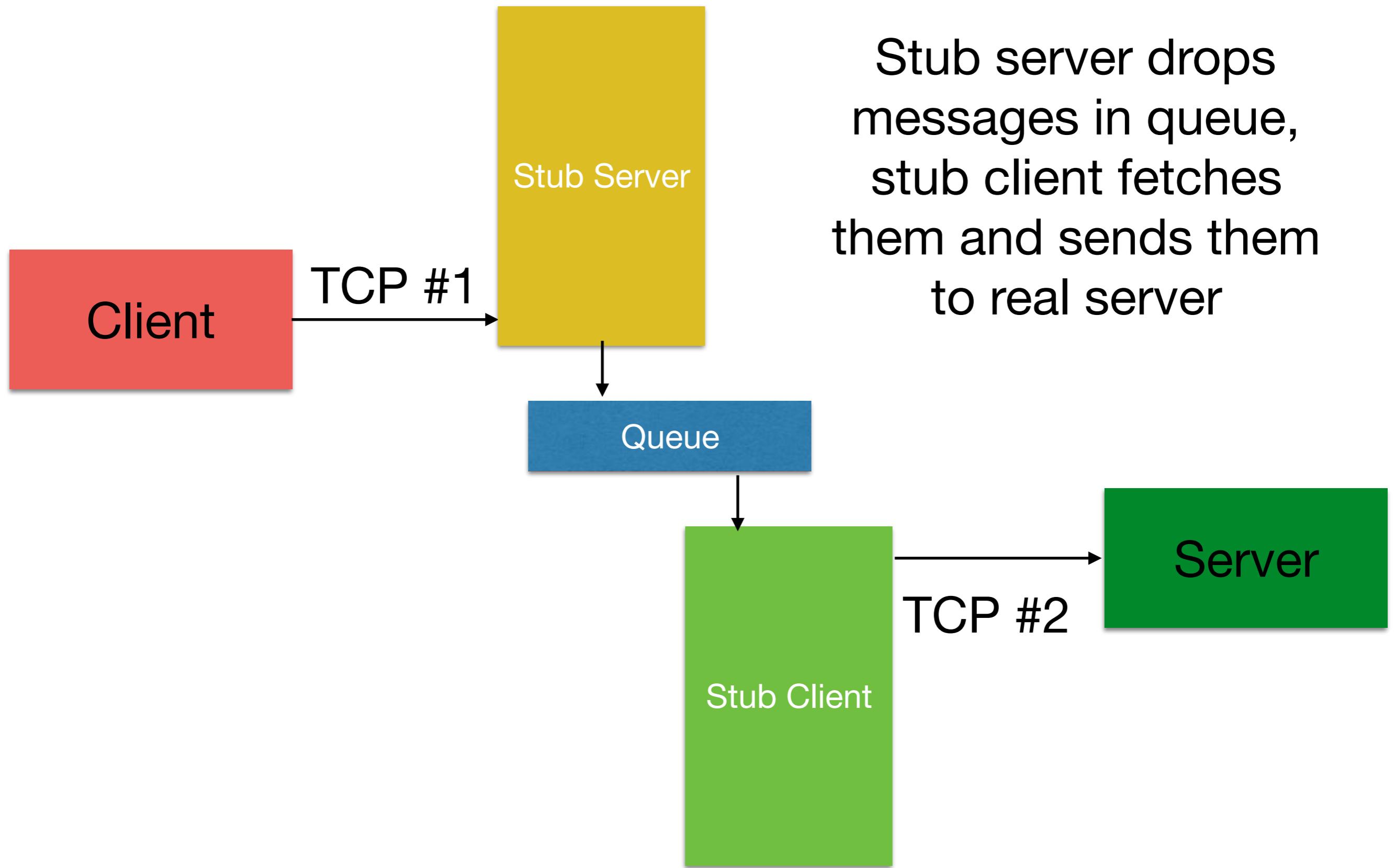
- Inbound proxy can also be used to unify namespace, so “www.my.dom.ain/some/URL” can be serviced on different machines for different URLs without exposing this fact to user.

# Mail

- Young people use IM, Twitter, Snapchat, etc.
- Business and government functions with email.
- So email is the bare minimum needed

# Mail Proxies

- Key point: mail is not real time
- Allows far more paranoid architecture



# Mail Gateways

- Any attack on internal server has to be embedded in messages
- But protocol is generated only by stub client
- Very hard to get chosen protocol operations through

# Mail precautions

- Often stub server has no list of addresses, so will accept mail to any address. Errors are sent back asynchronously.
- But can check addresses for correct format, to avoid embedding of buffer overruns (etc) into addresses.
- Can limit number of addresses, rate, etc, etc.

# Queue Precautions

- Queue can include virus scanning, etc.
- There are known attacks on virus scanners (for example, “compression bombs”)
- But scanner is not exposed to attacker, and has no path to either Internet or inside systems
- Bi-directional mail is usually done with two such set-ups, one in, out out.

# Data Diodes

- “Outside, Queue, Inside” can be generalised
- “Data Diodes” are a popular concept in multi-level secure (protectively marked) environments.
- Usually custom built, but basic idea is same: bring data to staging area, then transmit onwards
- In extreme cases, can require physical re-wiring (accept for an hour, forward, repeat)

# Voice

- Serious problem: voice is business critical and cannot be relayed with delay
- Has slowed adoption of voice over IP: keeping the voice in a completely separate infrastructure is attractive for security in buildings with existing cable plant.
  - And in 2017, who cares about fixed line telephony anyway?

# Session Border Controllers

- Phone registers with central call server
- SBC keeps track of registrations
- Allows incoming signalling from call server, with protocol inspection proxy-style
- Allows incoming media stream once signalling is in correct state
- Also implements (for example) lawful intercept, codec translation, dealing with NAT...

# VoIP Issues

- Of course, it's very difficult indeed to examine a stream of 8KHz, 8-bit samples and test to see if it is real voice or some other protocol being tunnelled.
- Hence very popular solution is to have VoIP on a separate VLAN, to keep traffic away from computers.

# VoIP issues

- For historical reasons, operators of VoIP networks are often liable to perform intercept when equivalent scenarios within non-voice ISPs would be carried out by law enforcement at law enforcement's expense.
- VoIP networks also need good overload control and 999/911 prioritisation.
- So SBCs are specialised, and able to name their own price.

# Ian's Prediction

- VoIP as a customer proposition is dead, because it's a replacement technology for analogue fixed line voice, which is also dead.
- Customers wanting fixed line voice may find it is handled with VoIP inside telco network, but it will be traditional POTS/ISDN to the customer.
- 4G voice is again VoIP inside telco network, but customer doesn't see this.
- So need to bring voice through main Internet gateway will reduce, not grow.
- **I could well be wrong.**

# Remember

- StuxNet was a suite of malware (probably) written by the US government, which targeted industrial controllers operating centrifuges.
- Centrifuges are the current best technology for separating fissile U235 from inert U238, to enrich Uranium either for nuclear power use or for weapons use.
- Stuxnet modified control programs to drive the centrifuges in an unstable way, to ruin their main bearings.

# Stuxnet

- The equipment in question was air gapped from the outside world: the only route to the controllers was via “sneaker net”: USB memory sticks, floppy drives, software updates on flash.
- It was still possible to infect the controllers.
- Proxies and Data Diodes protect against some network attacks, but are not perfect.

# Summary

- Proxies can protect against some outbound issues (DLP, downloading of viruses)
- Proxies hugely improve security on inbound services, especially when inbound servers are complex, privileged and old
- Data Diodes useful in classified environments
- Still can be bypassed by sufficiently resourceful attacker.

# Security 12: Wireless and Wired Security

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Wired Networks

- You know where the wires are (you hope)
- Tapping wires requires significant effort and is detectable
  - “Tempest” attacks are extremely difficult, although not impossible, as UTP and (more so) STP cable doesn’t radiate very much
  - Fibre available to reduce Tempest risk to essentially zero.
- Paranoid sites can use time-domain reflectometry (TDR) to find cuts and splices in cables and fibres, and optical power monitoring to find bends.
  - Cheap testers won’t find splices, but you pay more, you get more. You can find open circuit faults with a £600 TDR tester. Renting a Megger Teleflex SX to map splices is £600 a week.

# Wireless Networks

- None of the above
- Unlikely to be limited to your site, and certainly isn't limited to your site if attacker has high-gain antenna.
- In 2017 a wire only carries one client's worth of the site traffic, while a wireless network carries much more, so benefits of interception potentially greater



# And insiders gain too

- Switched wired networks make insider (people with legitimate access to the cable plant) attacks harder: they need to tap either more cables, or cables closer to the core of the network and therefore ideally more secure
- A wireless network offers traffic between multiple pairs of systems without leaving your desk: have receiver, have keys.

# Multiple problems

- Access control: don't want attackers on the network able to send and receive traffic as end-stations themselves
- Integrity: want packets to be unchanged in flight and to come from where they purport to come from
- Confidentiality: don't want attackers to be able to “sniff” traffic and read its contents
  - Note: not always necessary to be able to read contents to make changes useful to an attack

# Reality Check

- Wired networks have some physical security, but it isn't great.
- Most wired networks aren't proof against someone putting their own kit on the end of a wire.
- Most networks don't protect against physical attacks on the plant
  - Ideally routine TDR, cables in pressurised tubes, cables on walls, fibre for backhaul, etc.
  - Reality is dense mess under the floor

# So questions are:

- Is our wired network secure enough?
- Is our wireless network as secure as our wired network, or secure enough?
- “enough” is the product of a rigorous risk assessment or, perhaps, a damp finger in the air.

# Wired Network Security

- MAC filtering
  - Switch only accepts traffic from a specific set of MAC addresses on a specific port, or...
  - Switch / Network only recognises devices with specific MAC addresses

# MAC Filtering: Little Value

- Will stop some configuration errors, and might prevent users moving equipment around and making a mess of your patching records
- Low-skill attacker can change MAC address of attack system to match authorised system
- Even if not supported by ethernet card (in practice always is) this is standard part of virtualisation stack for bridging.

# Why do people MAC filter?

- Habit, or ritual
- Does make it easier to sack people, as they have to make an overt move in order to perform their attack (like having “No Trespassing” sign on fence). **This can be very useful.**
- Reveals assumption that controlling equipment controls capabilities, which isn’t really true

# 802.1x Authentication

- Client device supplies username and password or proves ownership of certificate (via various mechanisms) which unlocks port it is connected to
- Roughly same protocols as used to do wireless authentication, in fact
- Pre-authentication, port is either restricted to authentication only, or is connected to restricted VLAN.
- “Posture Analysis” allows running of checks on patching and virus scanner levels, with re-mediation

# Problems with 802.1x

- User interfaces for supplying credentials early in boot can be hostile
- Difficult to generalise to printers, servers, Web Cams...
- Are you authenticating the user (they can supply their username/password pair to any device, including their own) or the device (where do you keep the keys so hostile users can't borrow them and put them on their own phone)?
- In early implementations defeated by a switch at the user end of the cable (device 1 authenticates, port opens, device 2 can also access the network)
- More recent implementations also MAC Filter, but can be attacked with carefully timed swapping of MAC address while attack switch holds link up
- Look up “multiple supplicant support” to see how complex it is to deal with multiple devices all trying to authenticate via a simple switch
- TPMs help key management, but not much (sadly, a universal truth)

# Benefits of 802.1x

- Again, makes it easier to sack people by forcing misusers to perform “overt acts”.
- “Posture Analysis” can enforce patching in co-operative environments, although it is not obvious that the complexity is justified.
  - Probably better solved with good host patching regime, but possible that 802.1x (fashionable 2005–10) plus BYOD (fashionable 2011–date) overlap.

# Problems with 802.1x

- Trivially bypassed in many cases, and hard to assure an installation
- No encryption of payloads, so an attacker who can patch into cables is no worse off
- Key management for non-personal devices is tricky
- Key management for personal devices is tricky
- Personal view: very few implementations successful and on-going

# Wireless Networks

- First attempt, WEP: “Wired Equivalent Privacy”
- Intended to make a wireless network about as secure as a naive wired network

# WEP

- Everyone has a shared key (40, later 104, bits)
- A 24 bit initialisation vector is generated, so total 64 or 128 bits of key material
- It's used to encrypt packets, using the RC4 algorithm
- All stations with the key can see all packet contents: behaves like old-fashioned ethernet

# WEP is Broken

- WEP is trivially broken, and is an object lesson in why you should not attempt to roll your own crypto.
- Attacker can attempt to obtain key, or can attempt to obtain evolved key stream
- Additionally, generating IVs is very difficult in embedded devices

# RC4

- RC4 uses a key and an initialisation vector to generate a evolved keystream of bytes, which are XOR'd with the message to form cipher text.
- Means that you have two possible attacks: recovering the key, or recovering the evolved keystream

# Keystream

- Trivial to inject known packets into the network, watch them emerge encrypted by access point, and obtain portion of evolved keystream
- Generating unique initialisation vectors extremely difficult at power-on of embedded device (no hardware RNG, air turbulence in disk drives main source of randomness in computers, packet arrival times known to and potentially controlled by attacker)
- In many cases, IVs were drawn from very small space, so number of distinct keystreams was very small

# Details of Attacks

- Fluhrer, Mantin and Shamir recovers keys
  - Amazingly, RC4 = Ron’s Cipher 4, Ron = Ron Rivest. Shamir = Adi Shamir. So that’s the R and the S from RSA public encryption, here on opposite sides of the RC4-breaking game
  - Plus a variety of other “practical” attacks on weak use of RC4 (malleability attacks, in particular)
  - **WEP: Completely Broken**

# WPA

- Emergency interim fix to WEP, adding sequence numbers, better key mixing, integrity checks
- Idea was that it could run on same hardware as WEP, so only needed a firmware fix: doesn't involve box swapping
- Mostly of historical interest
- Still weak

# WPA2

- Proper encryption for grown-ups: third time lucky
- Initial secret (either PSK or result of per-user login) used to generate a transient key used just for one session (~1 hour)
- Underlying encryption is AES128
- Used correctly, can be accredited to IL4 CONFIDENTIAL in old money, possibly even low end of SECRET under new system (higher would require AES192).
- Needs new hardware
- You can't see other people's traffic (special extra key for broad/multicast traffic).
- Key management complex, risky and difficult to do correctly: using WPA2 for more than access control probably not for the casual user.

# WPA2 Authentication

- If you have a crypto background, look it up
- Basically, each party generates a random, sends it to other party, combines with shared key and generate another key. If the keys agree (ie, encrypt and decrypt identically) then both sides have the shared key and can communicate.

# WPA2 PSK

- Users share a common pre-shared key
- Key derivation function (PBKDF2) repeatedly iterates a strong hash function to generate better key material, maybe.
  - Mixes in SSID, so benefit to having an SSID no-one else is using is forces dictionary attacks to start afresh for you (random works well)
- Compromise of key exposes all past recorded traffic and all future traffic: no forward secrecy on session keys
- Compromise of key requires global change in order to lock out attacker who has old key
- Only usable for small, trusted groups of people (although in practice businesses use it more widely)
- Someone who captures a key exchange and has the pre-shared key can read traffic on that connection (special case of lack of forward secrecy)
  - Doesn't use Diffie-Hellman
- KRACK (allows resetting of connection, now mostly patched)

# WPA2 Enterprise

- Users authenticate individually to authentication server (usually using “Radius”) and a per-user key is negotiated (with some extra magic for broad- and multi-cast).
- Can interwork with SSO/AD/etc type systems
- Derived keys are per-user, so compromise of user key only compromises their traffic
- eduroam shows this working world wide, but...
- ...mostly for authentication, not confidentiality

# WPA2 Enterprise

- It's also worth noting that a lot of equipment doesn't support it, because it's tricky to implement and requires quite a lot of GUI support, testing and so on.
  - WiFi radios, televisions, set top boxes, gaming systems, etc, etc, etc.
  - This limits applicability in hotels, halls of residence, business parks / shared workspaces.
- There's also the issue that enrolment is awkward, as users need to be enrolled in whatever backend authentication system is being used.
  - This is true of so many things that "enrolment is awkward" should be a tee-shirt or something.

# Wireless Security Objectives

- Authentication
  - I don't want the wrong people using my network, I want to be able to identify people who are using my network, but if they need confidentiality or integrity that is their problem.
  - Hotel/Coffee/etc network
  - For practical purposes, “any network you do not personally control”.

# Confidentiality

- I want people using unencrypted applications to be able to work without being spied on by a wireless eavesdropper
- More corporate, where there are a lot of old applications which can't easily be SSL-ised
- I may not need forward secrecy, because I am not worried about people recording encrypted traffic to decrypt later if a key leaks (perhaps this is not a good assumption).

# Integrity

- I don't care if users' traffic can be read, but I don't want their traffic to be routed elsewhere (ie, make MITM attacks as hard as possible)
- Opens up issue of Secure DNS: attacker can provide fake addresses for known names, and then play games with redirects.

# WPA2

- Strong for authentication: no known attacks better than brute force (but see next slide)
- Worrying for confidentiality, because of lack of forward secrecy. Certainly, WPA+PSK is not sufficient for anything more than a family (small number of users, nothing wildly secret). WPA Enterprise better, but still not great, and complex.

# Warning: WPS

- Wireless Protected Setup
- “Press a button, get a key”, but also offers authentication using PINs
- Frighteningly insecure: should not be used, ideally should be disabled (although some systems disable it without actually disabling it)

# WPS Error

- If a PIN has eight digits, requires  $10^8/2$  guesses on average
- If system will verify PIN four digits at a time, requires  $10^4/2 + 10^4/2 = 10^4$  guesses.
- Reduces security from  $n$  to  $\text{sqrt}(n)$  operations
- And in fact, in the 8 digit PIN the last digit is a checksum, so  $10^4/2 + 10^3/2 = 5500$  guesses on average.
- Again, don't roll your own crypto...

# Again, in passing

- Generating a random number on an embedded device is difficult without hardware assist, especially close to power-on before serious traffic has passed.
- An attacker who can observe all traffic can probably massively reduce search space: where else is the entropy coming from?
- Pro tip: to sound informed in a security meeting, “where is the entropy coming from?” is always a good question.
  - Another tee-shirt.

# So, wireless security

- Option 1: use the security your wireless systems have
- Option 2: assume the wireless is open, use a VPN
- Option 3: option 1 for access control, option 2 for confidentiality and integrity

# VPN

- Details in a later lecture, but a VPN (“Virtual Private Network”) offers confidential communications over an untrusted carrier, using arbitrary authentication and encryption between end points.
  - Can easily offer full forward secrecy, two factor, etc
  - Extensively analysed products, mostly built on well-understood underpinnings (TLS, IPsec).
  - Available in trusted/accredited/etc versions up to SECRET and beyond (although these solutions are standalone hardware, as the keys can't be allowed into end-user computers).
- VPNs offer high confidentiality and high integrity, provided you are not too bothered about availability: you can always jam wireless given enough resources, and frequency-agile and/or spread-spectrum jam-resistant networking requires exotic, exotic licensing you can't get unless you are military.

# Rogue APs

- Common meme: “don’t do banking in hotels”. “Don’t do banking in Starbucks”.
- Evidence for efficacy of these attacks at scale is not great: they rely on poor user behaviour and are quite complex to execute.
- But the place to defend against this is in browsers and in websites; attempting to make all public WiFi secure is impossible.
- And there are other benefits: makes compromise of home/workplace WiFi less serious if assumption is that it is broken anyway.

# Rogue AP

- Option 1: assume users access bank via Google search for name. Catch lookups of Google A records or TCP connections to known Google addresses, hand back results with fake URLs for bank (110ydsbank, that sort of thing).
  - Protections include https, certificate pinning, user education, EV certificates, etc.
  - Harder (but not always much harder) to pull off since Google went https, depending on browser behaviour.
  - I have seen some hotel/shop networks which try to force you to use unencrypted Google, presumably as part of doomed content-filtering mechanism.

# Rogue AP

- Option 2: Fake DNS records for mybank.co.uk
  - use it to serve fake certificate and hope user does not notice; or
  - use it to push user to site which redirects to mybank.co (or something else visually similar) for which you have a real certificate; or
  - use it to push to site which redirects back to unencrypted version and hope user does not notice.
  - Will be caught by certificate pinning, HSTS and so on.

# Remember

- We often dismiss potential attacks on the grounds that skilled opponents would not be caught by them.
- However, attacker does not need to get lucky every time, or even most times, they just need to get lucky occasionally.
- As people who notice and avoid the attack don't have an obvious way to report it, or share it with less skilled users, the attacker can keep on trying: the failed attempts cost them nothing.
- If you went to the barista in a Costa and reported that there was something awry with the wireless network, what do you think would happen? My guess is “nothing”. Would you have the energy to take it further?
- All that said...

# Rogue AP

- Much discussed, not much evidence of wide deployment against passers by.
- Probably used for targeted and focused attackers who know what they are doing and don't get caught.
- Absence of cryptographic protection on initial DNS lookups very scary, probably more exploits to come
- Good argument for VPN: if you're using a wireless network you don't trust, connect to a VPN which you do trust (I use [privatetunnel.net](http://privatetunnel.net), as well as my home server).

# Public Service Announcement (1)

- HSTS: HTTP Strict Transport Security, **RFC 6797**
- HTTPS header which says “for the next XXX seconds (*31536000, one year, is good*), this website will always be HTTPS, and you should reject any attempt to serve it **unencrypted**. Cache this fact”. Means attacker must catch **first** access to site by browser installation/user pair.
- HSTS Preload into Chrome and derivatives mean attacker can never force use of unencrypted version.
- **Please, please, please convince your managers to use this.**
- **add\_header Strict-Transport-Security "max-age=31536000";**

# Public Service Announcement (2)

- HPKP: HTTP Public Key Pinning **RFC 7469**
- HTTP header which says “this site will always use one of these public keys, until at least this time; cache this fact”. Stops fake certificates stone dead.
- Tricky to set up as requires you advertise future public keys in case of revocation/compromise, hence need to keep offline copies of private keys. Could be tricky if hardware storage modules are involved. Possible DoS issues (recovery from compromise is hard).
- In 2018, looks dead: not going to be implemented in Edge or Safari, isn’t implemented in Firefox, is slated for removal in Chrome over next few months.

# So...

- Use wireless authentication to keep out people you don't want using the infrastructure, but don't trust it for confidentiality (no forward secrecy, probably no one-time passwords) and encourage users to take their own precautions based on their personal risk appetite.
- Layer a VPN over the top to provide desired level of confidentiality
- Proper VPN boxes are much cheaper than they used to be.

# And of course...

- If you're able to justify running a full-on VPN on the wireless network, you have to ask whether you should do something on the wired side as well.
- IPsec is a later lecture

# Summary

- Wired networks have some security, but you can add more; 802.1x isn't great, but there's IPsec or VPN technologies if you really need it.
- Wireless networks are wide open, or at least best treated as such, and if you're doing anything of even passing confidentiality you need either IPsec/VPN or to be confident in the encryption of your traffic.
- IPsec and other VPNs are well analysed and available in assured/accredited forms.

# Network Security 13: IDS and Apps

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Recap on Firewalls

- Firewalls look at packet headers and block/pass based on protocols, sources and destinations
- Help implement policy on access to applications and keep out random probing, but have limited benefits for well-run networks

# Recap on Proxies

- Terminate connections, process through protection mechanism, run new connection to service
- Various approaches for encryption
- Protect against protocol-level attacks
- Require complex new code for each new protocol, or alternatively a pass-through which misses the point
  - Look up SOCKS5

# Recap on Malware Protection

- Multiple means of operation:
  - Signature-based, looking for patterns in files
  - Heuristic-based, looking for virus-like behaviour in files
  - Behaviour-based, monitoring the actual behaviour of running programs (cf. Apple sandboxes)

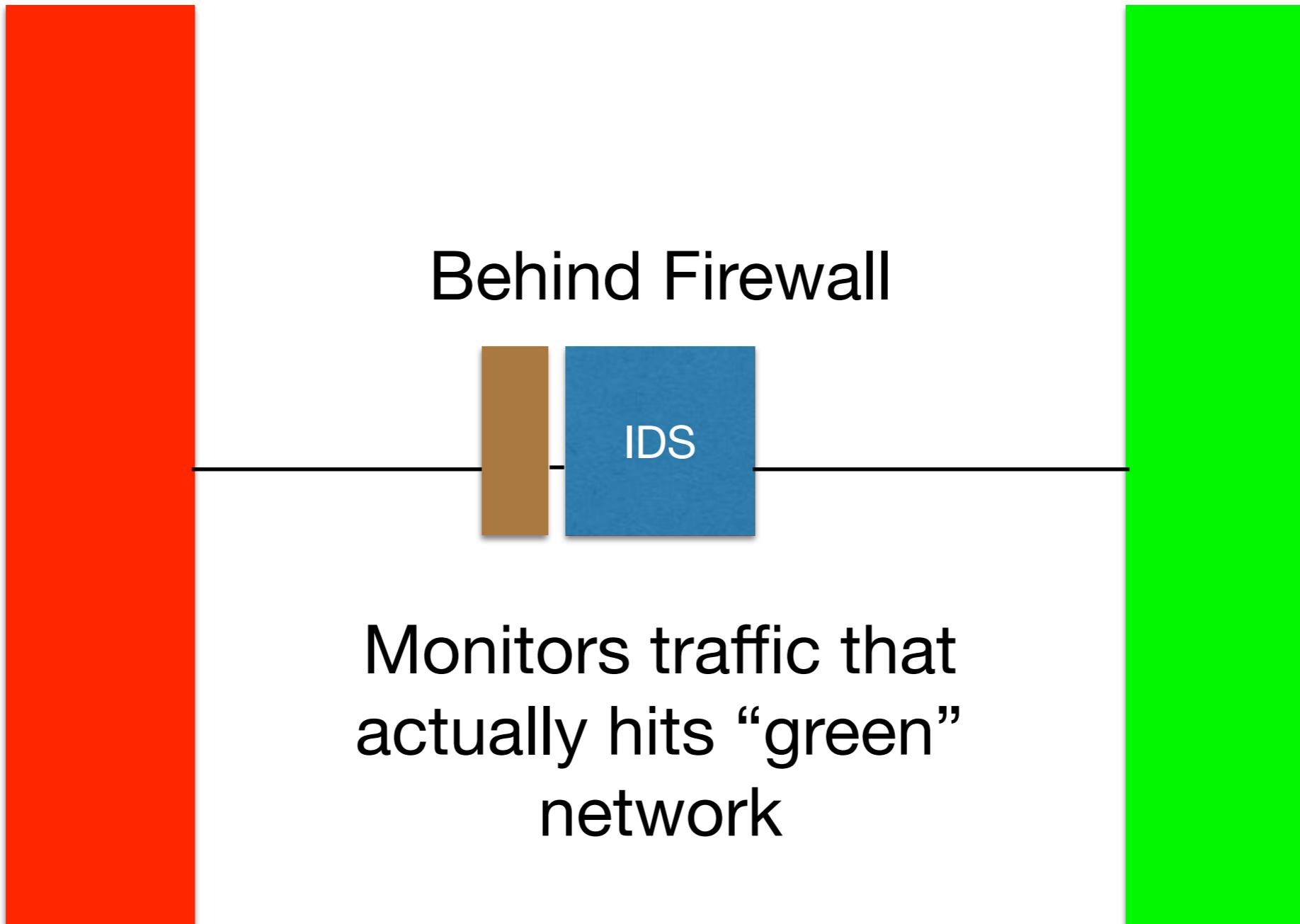
# IDS Concept

- Intrusion Detection System
- Pass all traffic through a system which treats network flows as target for malware analysis
- Has access to headers, payloads, timing, source and destination...

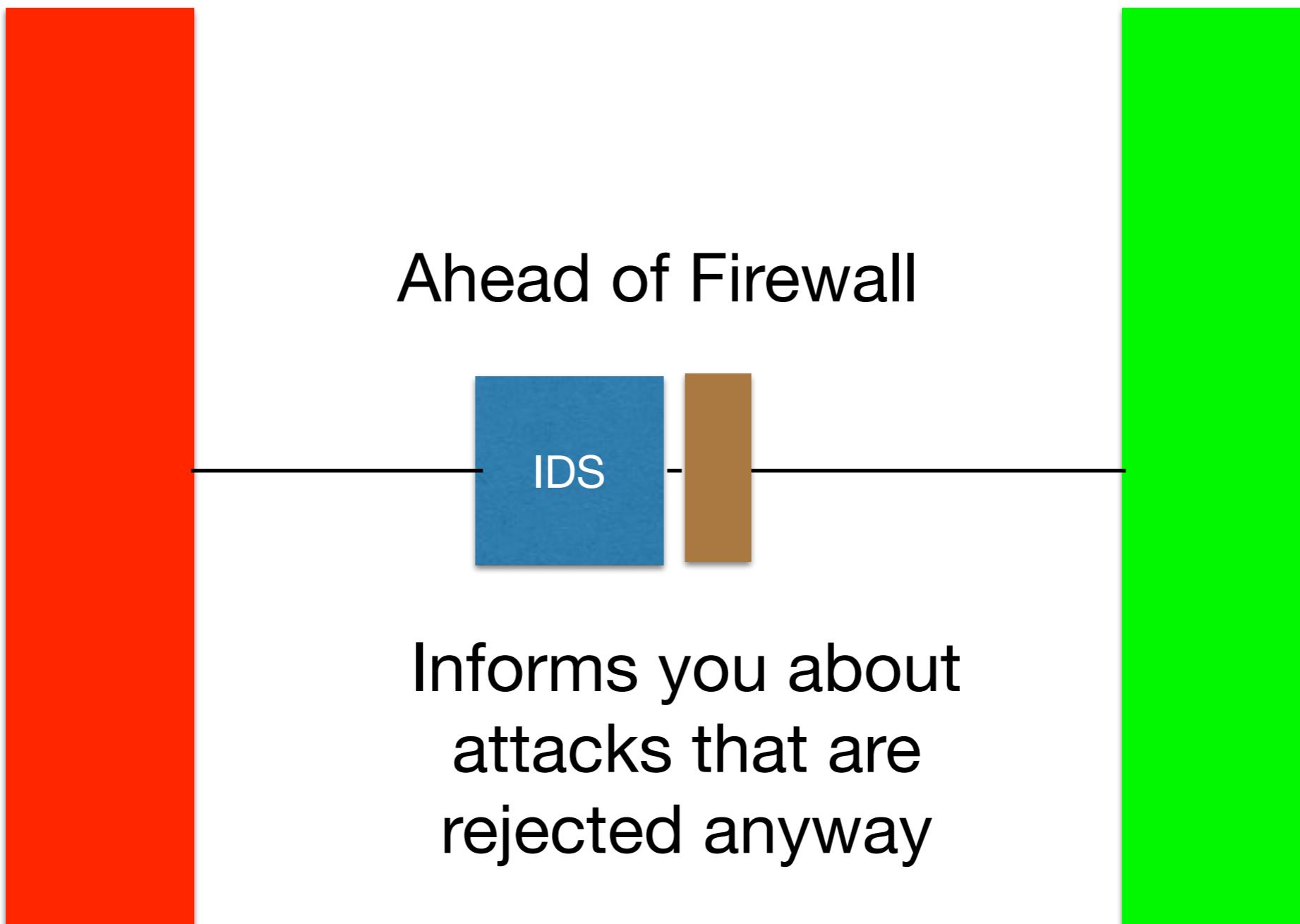
# IDS Triggers

- Patterns in packets: matching payloads of packets against signatures of known malware or other attacks
- Behaviour: matching against known patterns of malware behaviour (probe this, probe that, report to the other)
- Heuristic: matching against changes in network performance, or obviously dubious activity (lots of failed connections, say)

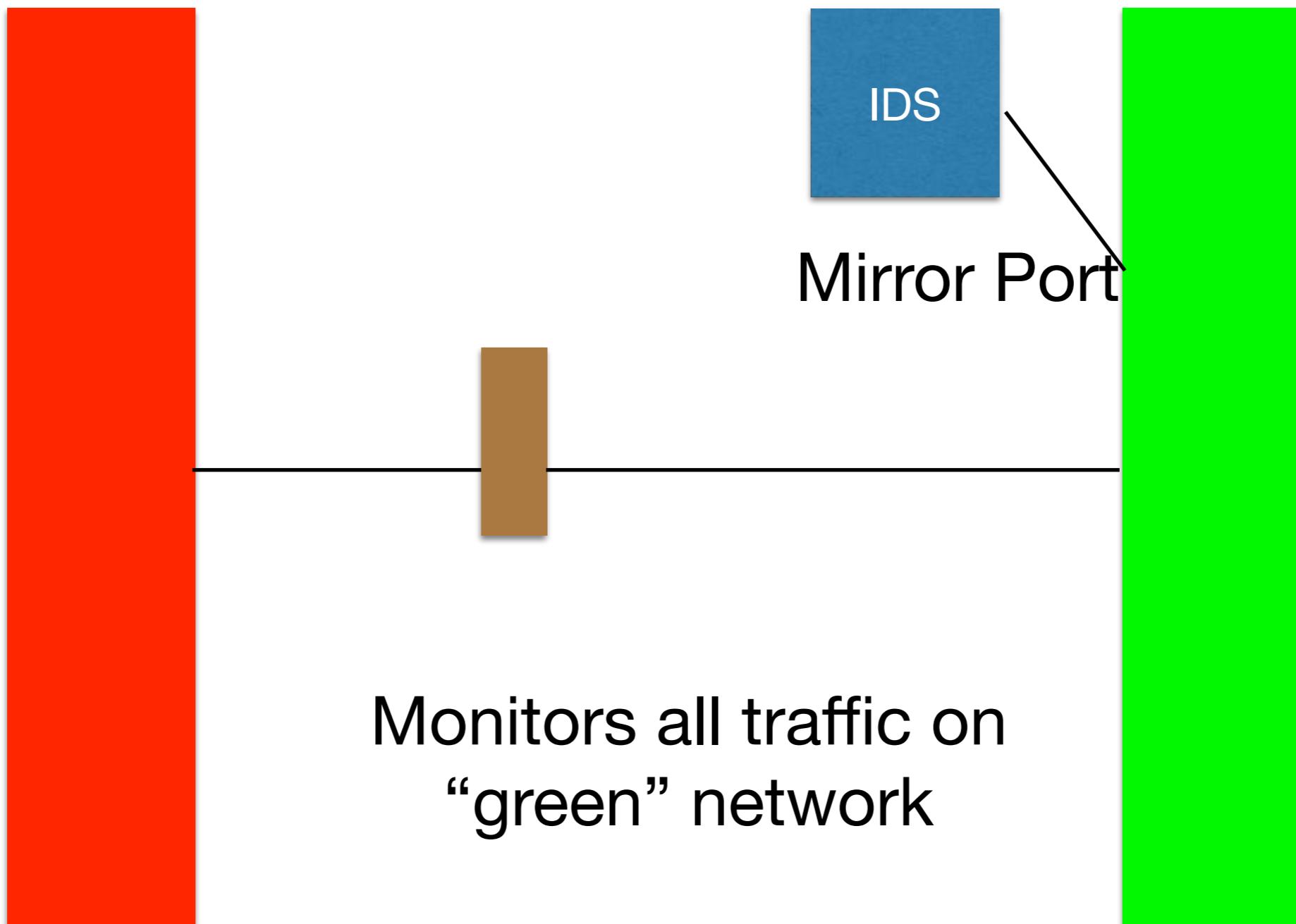
# Obvious Deployment



# Research Deployment



# Ideal Deployment



# Usually “Mirror”

- Typical switch has facility for “mirror” port that receives traffic to and from all other interfaces
- Meant for debugging and monitoring
- Obvious issues over performance: some buffering to cope with bursts, but will drop traffic under load
  - Switches rarely have deep buffering, and mirror ports don’t normally have extra buffers

# IDS: Drinking from Firehose

- IDS therefore receives all traffic, including unicast between stations on network
- Modern fast processor can handle >1Gbps stream with fairly complex matching, 10Gbps with multicore
- Architectures are often multi-stage, to discard least interesting stuff as early as possible

# IDS Problems

- Match packets against range of threats
- Lots of false positives
- Many attacks old / patched / irrelevant to current infrastructure
- Heuristics against zero-day attacks triggered by new software of many types
- Highly unlikely to detect spear-phishing
- At best, another line of defence against rapidly-spreading malware and (if you are very lucky) zero-day buffer-overrun type attacks
- "We think IDS is dead. It's failed to provide enterprise value," [Gartner] says. "In order for it to survive, it has to go faster, at wire speed, and it has to solve the false-alarm problem."

# Sample IDS Rule

```
alert tcp any any -> any 21 \
  (flow:to_server,established; \
content:"root"; pcre:"/user\s+root/i";)
```

Two-stage rule: match “root” (fast) then confirm with regular expression

# IDS Rules

- If your FTP server accepts root logins which really get root, you have massive problems, and spotting the attempts in network streams is no substitute for proper audit
  - FTP servers should run chroot() under all circumstances
- But if your FTP server doesn't accept root logins, why do you care if someone tries?
  - Might be a sign of reconnaissance by attacker, but are skilled attackers really so obvious?
- Provides way for real attackers to drown you in noise

# IDS Rules

- Lots of rules are fragments of known attacks, often buffer overruns and the like
- Trivially easy for attackers to mutate
- If you have time to update rules file, you have time to patch or mitigate the actual problem

# IDS->IPS

- Can be in-line (responds faster) or mirror port (easier to deploy)
- Re-programs firewalls or switches to close off attack
- All the same false positive problems, but now automated
  - Less work, or more trouble?

# Lots of products

- Snort (free)
- Tipping Point (\$\$\$)
- Many, many others
- I would be very interested to see a mature, site-wide, well-tuned implementation
  - Suspect they're turned off before they are finished, but welcome counter-examples

# Research Topics

- Machine learning on protocols, traffic patterns, etc
- Spot “normal” behaviour, trigger on “abnormal”
  - Probably involves so much hysteresis that response time would be in hours
- Lots of research, lots of adverts, not a lot of big deployments
- Plenty of potential for a PhD

# Research Topics

- Legal and ethical issues over scanning all packets
  - Inside an enterprise it's all OK
  - Still some debate about legality of ISPs scanning email for viruses and spam
  - “Deep Packet Inspection” at ISP level relies on fine distinctions about whether algorithms constitute “reading”.
- Machine learning based on S/Flow data very interesting

# HIDS

- Host Intrusion Detection
  - Tripwire
  - OSSEC

# Tripwire

- Take MD5 / SHA1 hashes of critical files
- Compare them with files on disk periodically
- Many technical problems (how do you secure the hashes, the scanner, the kernel...?)
  - VM introspection looks interesting
- But useful against naive attacks

# Using Tripwire

- Boot off secure installation (CD, for example)
- Do tricky things with multi-tenant storage, NAS, etc
- Protects against some insider threats
- Also against careless installation procedures that overwrite system files

# OSSEC / HIDS

- Host Intrusion Detection
- Originally developed by Trend (tier-2 virus scanner vendor) as open source with their support
- Now open source without their support (pretty much)

# OSSEC

- Tripwire, plus log analysis, plus filtering and response
- Can, for example, update firewall after multiple bad login attempts
- Works best with live feed of logging information

# OSSEC Rules

```
<rule id="5702" level="5">
  <if_sid>5700</if_sid>
  <match>^reverse mapping</match>
  <regex>failed - POSSIBLE BREAK</regex>
  <description>Reverse lookup error (bad ISP or attack).</description>
</rule>

<rule id="5703" level="10" frequency="4" timeframe="360">
  <if_matched_sid>5702</if_matched_sid>
  <description>Possible breakin attempt </description>
  <description>(high number of reverse lookup errors).</description>
</rule>
```

# Host Intrusion Works, ish

- Knows the platform it's on, so focuses on attacks that are relevant (ie, not looking for Windows attacks against Unix platforms)
- Sat behind all firewalls and other access control, so only looking at abnormal events
  - Downside: has less information to work with
- Running it on small networks yields low levels of false positives after minor amounts of tuning, so might be worthwhile...
  - ...except has never detected anything terribly serious either...

# HIDS Problems

- Automated log-reading is very tricky, and small changes to software can break it
- Small changes to messages seen as dangerous can cause them to be ignored
- You can end up needing to match everything and raising all unexpected log messages as attacks
- And we're back to the false positive / noise debate

# fail2ban, etc

- HIDS can be used to detect repeated failure of login or other access attempts, and then program firewalls to block them
- Simpler software (tcp\_wrappers, fail2ban, built-in features in firewalls and daemons) can do the job just as well
- Not clear that stopping repeated login attempts at network level prevents break-ins
  - Rate limiting on individual users is good, but most such probing tries varying username/password combinations
  - Can provide mechanism for DoS attack (do not lock accounts based on repeated login attempts unless you really understand the risks)
- However, will reduce noise in logs.

# Firewall Friendliness

- Applications will need to live behind firewalls, IDSes and so on
- It's important they behave sensibly and securely

# Single connection

- Where possible, applications should work over a single TCP connection (performance, security, firewall states)
  - Starting multiple encrypted connections is particularly expensive
- If they need multiple connections (IMAP is an example), each one should go through the full authentication protocol or the mechanism should be very carefully examined by an expert

# Client always initiates, server always listens

- The client application should be the only party to call `connect()`, the server should be the only party to call `listen()` and `accept()`
- Otherwise doesn't work via NAT or through a firewall

# No port numbers or addresses in payload

- Protocols should never pass address information within their payload
  - Probably implies you're going to break rules on previous page
  - Doesn't get NAT'd, doesn't work well with firewalls

# Logging

- Despite all my scepticism about IDS/HIDS/etc, it is vital that as much network activity as possible is logged (yes, I realise “as much...as possible” is a somewhat open statement).
- **Forensic Readiness** is next year’s hot topic. Even if you don’t know what to do with the logs, a forensic team might find them useful.

# Logging

- Logging is one of the advantages of running network proxies: you can centralise logs of activities in places which both attackers and local users cannot easily manipulate.

# Security 14: TLS, OTP

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Recap

- Attackers can intercept packets
- Attackers can modify packets
- Attackers can inject packets
- Attackers can discard packets
- “A Dolev Yao attacker”

# TLS

- “Transport Layer Security”
- Standardisation and extension of Netscape’s “SSL” Secure Sockets Layer.
- Idea is to provide an encrypted layer over TCP, so that applications that run over TCP will easily run over TLS and get all of TCP’s properties and CIA.
- Lots of security implications and, as ever, crypto is hard to do right.

# TLS has lots of options

- Basic idea is that we use:
  - asymmetric encryption to agree a key,
  - some sort of zero knowledge proof to show possession of private key for certificate (might be combined with previous stage)
  - symmetric encryption to encrypt data
    - with HMAC or hash to prove integrity
- Some combinations are weak, and implementations have flaws
- TLS supports huge range of negotiated cipher suites.

# Want to bet on all of these?

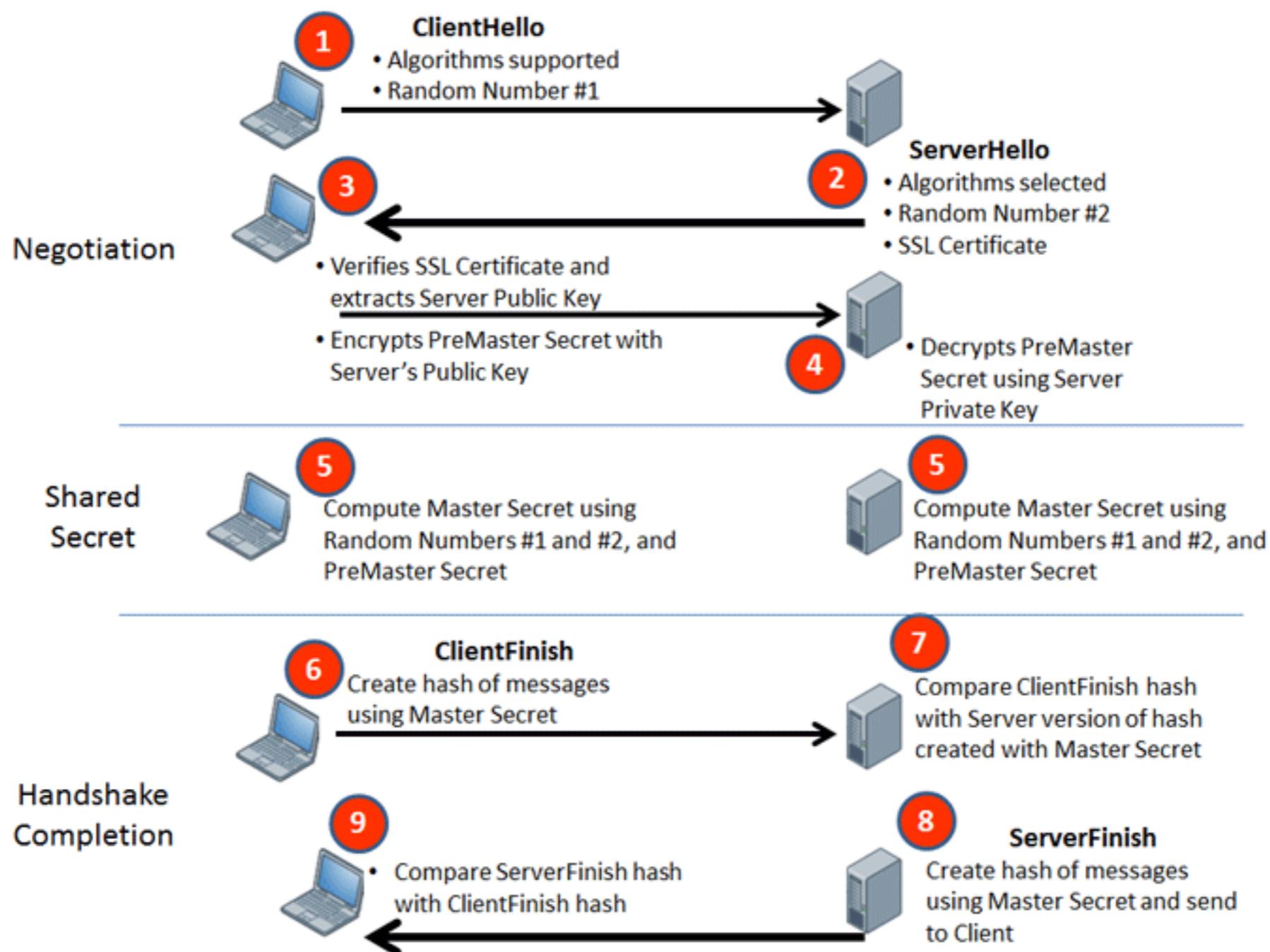
```
ians-macbook-air:NETSEC16 igb$ openssl ciphers | tr ':' '' | sort | pr -t -4 -w 132 | col -x
AES128-GCM-SHA256          DH-RSA-CAMELLIA128-SHA
AES128-SHA                  DH-RSA-CAMELLIA256-SHA
AES128-SHA256                DH-RSA-DES-CBC3-SHA
AES256-GCM-SHA384           DH-RSA-SEED-SHA
AES256-SHA                  DHE-DSS-AES128-GCM-SHA256
AES256-SHA256                DHE-DSS-AES128-SHA
CAMELLIA128-SHA             DHE-DSS-AES128-SHA256
CAMELLIA256-SHA             DHE-DSS-AES256-GCM-SHA384
DES-CBC3-SHA                DHE-DSS-AES256-SHA
DH-DSS-AES128-GCM-SHA256    DHE-DSS-AES256-SHA256
DH-DSS-AES128-SHA           DHE-DSS-CAMELLIA128-SHA
DH-DSS-AES128-SHA256        DHE-DSS-CAMELLIA256-SHA
DH-DSS-AES256-GCM-SHA384    DHE-DSS-SEED-SHA
DH-DSS-AES256-SHA           DHE-RSA-AES128-GCM-SHA256
DH-DSS-AES256-SHA256        DHE-RSA-AES128-SHA
DH-DSS-CAMELLIA128-SHA      DHE-RSA-AES128-SHA256
DH-DSS-CAMELLIA256-SHA      DHE-RSA-AES256-GCM-SHA384
DH-DSS-DES-CBC3-SHA         DHE-RSA-AES256-SHA
DH-DSS-SEED-SHA             DHE-RSA-AES256-SHA256
DH-RSA-AES128-GCM-SHA256    DHE-RSA-CAMELLIA128-SHA
DH-RSA-AES128-SHA           DHE-RSA-CAMELLIA256-SHA
DH-RSA-AES128-SHA256        DHE-RSA-SEED-SHA
DH-RSA-AES256-GCM-SHA384    ECDH-ECDSA-AES128-GCM-SHA256
DH-RSA-AES256-SHA           ECDH-ECDSA-AES128-SHA
DH-RSA-AES256-SHA256        ECDH-ECDSA-AES128-SHA256
ECDH-ECDSA-AES128-SHA      ECDH-ECDSA-AES128-SHA256
ECDH-ECDSA-AES128-SHA256   ECDH-ECDSA-AES128-SHA256
ECDH-ECDSA-AES128-SHA256   ECDH-ECDSA-AES128-SHA256
ECDH-ECDSA-AES128-SHA256   ECDH-ECDSA-AES128-SHA256
```

```
ECDH-ECDSA-AES256-GCM-SHA384
ECDH-ECDSA-AES256-SHA
ECDH-ECDSA-AES256-SHA384
ECDH-ECDSA-DES-CBC3-SHA
ECDH-ECDSA-RC4-SHA
ECDH-RSA-AES128-GCM-SHA256
ECDH-RSA-AES128-SHA
ECDH-RSA-AES128-SHA256
ECDH-RSA-AES256-GCM-SHA384
ECDH-RSA-AES256-SHA
ECDH-RSA-AES256-SHA384
ECDH-RSA-DES-CBC3-SHA
ECDH-RSA-RC4-SHA
RC4-MD5
RC4-SHA
SEED-SHA
SRP-3DES-EDE-CBC-SHA
SRP-AES-128-CBC-SHA
SRP-AES-256-CBC-SHA
SRP-DSS-3DES-EDE-CBC-SHA
SRP-DSS-AES-128-CBC-SHA
SRP-DSS-AES-256-CBC-SHA
SRP-RSA-3DES-EDE-CBC-SHA
SRP-RSA-AES-128-CBC-SHA
SRP-RSA-AES-256-CBC-SHA
```

# Certificates

- A public key, signed by a certification authority
- I send you a certificate
- You check the signature (compare hash of certificate with signed value, using known CA key)
- You send me a random, encrypted with my public key
- I decode the random and do something to prove knowledge of it (send back the random in clear, use it to encrypt a known string, etc, etc)

# Sample handshake



# RSA key establishment

- Client generates a random number
- Encrypts it with server's public key
- Sends result to server
- Both parties now share the random and can generate a session key from it
- Also proves server has private key matching public key

# Problem #1

- Clients are rubbish at generating random numbers
  - Without hardware RNG, best source of randomness is turbulence in disk drives
  - So useless for portable devices
- Sadly, not fixable with current protocols, but feeds into the more serious problem...

# Problem #2

- Communication starts with session key (or material used to make session key) encrypted with server public key
- Server public key (therefore also private key) is used long-term (potentially years) because certificates have long lifetimes
- Therefore **later** compromise (including warranted) of private key reveals session keys for all previous sessions, if they have been intercepted
- Problem is referred to as **forward secrecy**

# Partial Solution

- Diffie-Hellman key exchange
- Pre-agree prime  $p$  and generator  $g$ 
  - Each party generates a random
  - Sends  $g^x \bmod p$ ,  $g^y \bmod p$
  - Computes  $(g^x)^y \bmod p$  and  $(g^y)^x \bmod p$
- Both parties now share  $g^{xy} \bmod p$

# Diffie-Hellman

- Usually done with elliptic curves, not integers mod p.
- Only as strong as the weakest random number generator, sadly
- Doesn't provide any proof of knowledge of long-term keys
  - So RSA is used with the certificate to prove key ownership

# Lots of TLS Attacks

- Lucky 13
- BEAST
- POODLE
- Heartbleed
- Some attacks on crypto, some on implementation
- Complex protocol plus complex implementation plus complex crypto = lots of bugs
- “Downgrade attacks” also difficult to deal with: convince client that server only offers weak crypto, or vice versa.

# Main implementation is a mess

- Most popular implementation is OpenSSL: has portability hacks for ancient, long-dead operating systems.
- Read up on LibreSSL and “Open SSL Rampage” for more details
- Read up on Debian SSL Random Number Generator bug CVE-2008-0166 to see myth of open source

# Waterloo TCP supported in OpenSSL, last touched 15 years ago, TCP/IP for DOS and Windows 3.1

This page contains my port of Waterloo tcp/ip (WatTCP). Watt-32 is an enhanced version of Geof Cooper's [TinyTCP](#) and [Erick Engelke's](#) WatTCP. The latest [version](#) is dated November 1999 with features integrated into Watt-32.

Watt-32 is a library for making networked TCP/IP programs in the language of C and C++ under DOS and Windows-NT. Both 16-bit real-mode and 32-bit protected-mode is supported.

For DOS, Watt-32 requires a packet-driver (*PKTDRVR*) to access the data-link layer (Ether-PPP, SLIP or Ethernet. Token-Ring is un-tested). With the correct packet-driver, it will run under *all* versions of Windows too. I highly recommend [SwsVpkt](#) which works much faster than Dan Lanciani's [NDIS3PKT](#). With the SwsVpkt or NDIS3PKT drivers one can connect to Windows services (on the same machine) from a DOS-box too.

For Windows, [WinPcap](#) and *NPF.SYS* are required. Note that Windows 95, 98 and ME are not supported.

The name Watt-32 was chosen to signal the emphasis on 32-bit platforms (Although 16-bit compiler are also supported). What, besides embedded systems, is DOS good for these days if not running high-performance 32-bit programs. And the embedded market is booming; with the price of PC-104/Ethernet cards, Watt-32 could be used in a lot of fancy boxes. How about an *IP-telephone*, *MP3 home-player* or an *Internet Radio*?

# Alternatives

- GnuTLS: has history of bugs almost as bad as OpenSSL
- PolarSSL: used in embedded systems, less bad history, but much less well tested
- LibreSSL looking better and better twelve months in, so probably now the go-to implementation

# HSMs

- Biggest problem is ensuring security of private key associated with certificate for domain
- Hardware Security Modules can store a key and perform only the appropriate tasks with it (use it to sign / decrypt but not release it)
- Could use a TPM but there are practical problems
- Unfortunately, CAs use HSMs, few other people do: more commonly a file in /etc...

# TLS Benefits

- Protects against passive attackers observing valuable traffic (TLS offers **confidentiality**)
- Protects against active attackers modifying valuable traffic (TLS offers **integrity**)
- Protects against man-in-the-middle attacks if certificates are properly checked (if, **if**, **IF**).

# TLS Problems

- Avoids all content filtering and checking services
  - Solutions to this worse than the problem
- Exceptionally reliant on correct issuing of certificates
- Typical “geek” security measure: provides good properties when implemented by experts and used by trained people, but full of pitfalls for “ordinary” developers and particularly for untrained users
  - “Write a guide for your grandmother to allow her to accurately check that a site has a correct certificate” is a good exam question, but not one I’d want to mark.
- This isn’t a security usability course, but TLS is one of the worst areas for usable security.

# Certificate Issuance

- Certificate Transparency: all CAs log the certificates they issue into a public log, so that you can
  - Check that certificates were properly issued
  - Look for people mis-issuing “your” certificate
- Certificate Authority Authorisation: you can declare in your DNS who is allowed to issue certificates for you, and all CAs must check
  - However, you can simply not do this, which means “anyone can issue”, and so far no-one (fsvo no-one) is doing it.
  - And most people don’t do DNSsec

# TLS Usage

- Can either start up immediately, and then have application run over the top (ie, connect to port 993, negotiate TLS session, then start IMAP)
- Or connect to port 143, talk unencrypted IMAP, then say STARTTLS and switch to encrypted communications, over the same TCP connection, still over port 143.
- Nicer for firewalls, as can see early stages, and one port better than two.

# Authentication over Networks

- What are we using TLS to protect?
- Often, just login details
  - For most people, it is much more important that people cannot create messages from you than confidentiality of messages to you
  - I'd rather people didn't read (or indeed modify) my bank statements, but the harm is small; I really don't want people spending money as me.
    - Yes, exotic attacks possible involving consistently changing bank statements to conceal other fraud: much more effective with malware, as relies on seeing all traffic for all sessions.

# TLS Authentication

- Client-side certificates are rarely used, which is a shame (I have only ever used one once, for a CA which – ironically – is now on Mozilla's final warning list)
- Stored protected either by OS facilities by a static passphrase
- Passphrases can be obtained with cameras, key-loggers, malware, etc, but obtaining the private key requires malware (probably main attack vector)
- Appeal is that the only material stored on server is a public key, whose theft is less serious than (say) a password hash.
- Enrolment is hard (per-app? per-server? per-user, but do I trust other issuers? What issuers?)
- Sadly, adoption is non-existent

# One Time Passwords

- Basic principle:
  - Secure piece of hardware (“token”) shares a secret with a server
  - Token combines (hash, encryption) secret with counter or clock, and displays the result
  - Used as password: server can check as it knows counter/clock and secret (crucially: it **knows the secret**, not a hash of the secret)

# Implementation Details

- Button might have been pushed off-line, or clocks might have drifted
- Accept the expected password plus the next  $n$ , and update on server the expected offset
- Can add PIN feature on token to either unlock it, or combine PIN into hash as well.

# Benefit

- Token is never connected to computer (or only connected “one-way” – Yubikeys pretend to be a keyboard)
- Secret cannot (fsvo cannot) be extracted, so token cannot be copied
- So long as server is secure, requires possession of token to log in

# Problems

- Expensive (even \$10 per token is expensive in large deployment) although soft-tokens help if security acceptable
  - Soft tokens rely heavily on security of app data on phones, good in iPhone  $\geq$  5S, somewhat random on Android.
- Server full of plaintext secrets is a major target
- Difficult to share between domains, so you end up with a fistful of tokens (worn around neck by geeks)
- Not hard to use, but not easy to use either

# One Time Over the Air

- You attempt to log in as “igb”
- System texts password to phone number registered to “igb”
- igb logs in using one-time password
- Requires possession of phone (or, more precisely, SIM private key — see recent news stories about GCHQ attacks on Gemalto) or control of GSM network.

# Benefits

- Phones are essentially universal amongst people likely to be using such services
- Solves problem of needing lots of tokens
- SIM security is well proven (GCHQ attacks on card issuers in a way proves how good the security is, as if security weak they could break it anyway)

# Problems

- SMS messages require infrastructure in the data centre
- Roaming costs
- Delay in transmission can be “just bad enough” to make usability poor (especially if roaming)
- SS7 and other parts of GSM infrastructure a cesspit: CESG/NSCS advice is to not do this, and certainly not for anything protectively marked.

# Popular for...

- Either as text or as voice call speaking the number, popular for authentication of new instructions to bank or (I found yesterday) setting up Apple Pay.
- Probably good enough for this purpose, but not for anything more serious

# Protects Against

- Key loggers (different password each time)
  - Hence cameras, malware, etc
  - Disclosure to others, sort of...
  - Requires physical possession of tag
  - Not resistant to shared use by phone

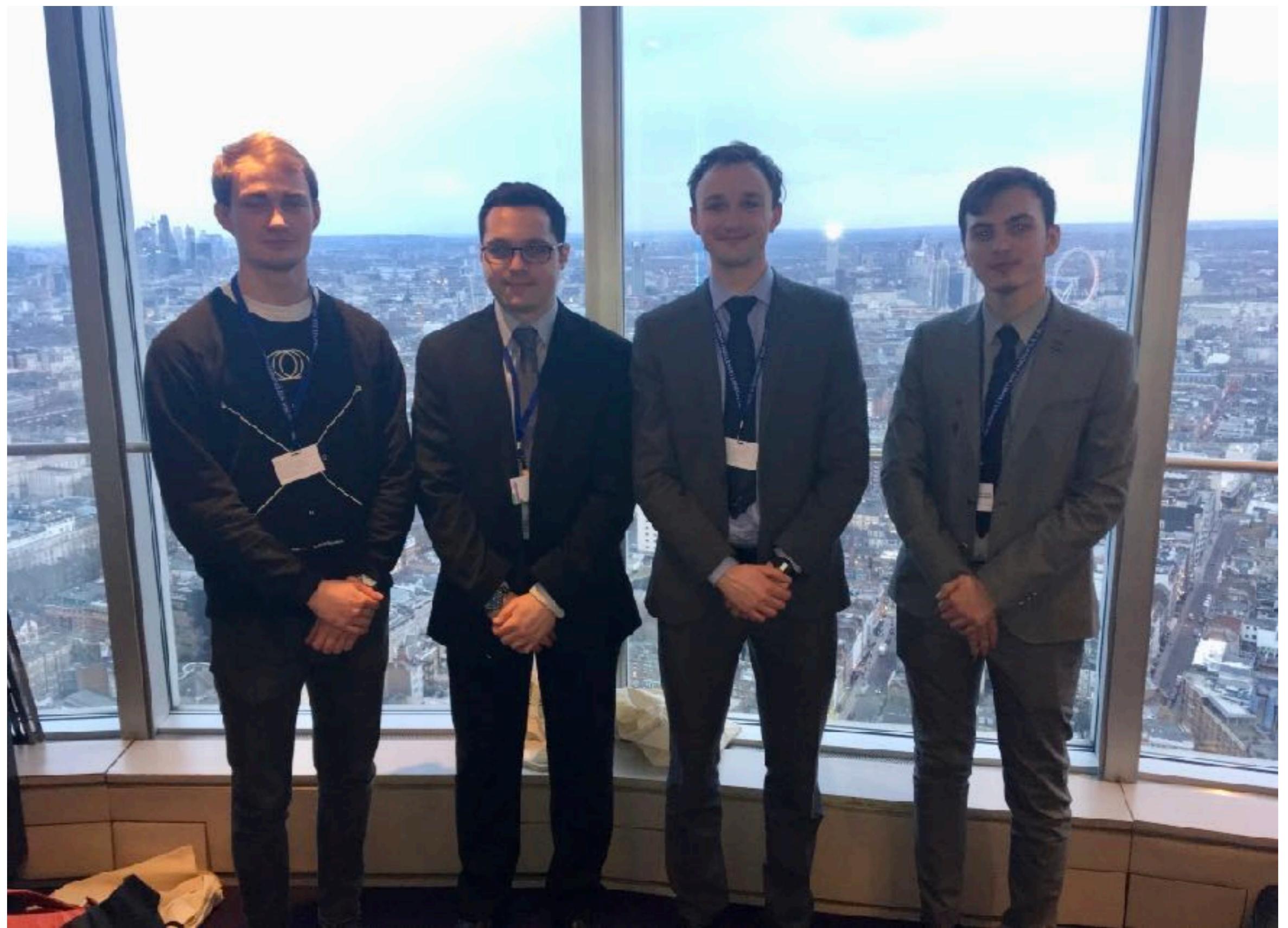
# Breaks

- Original product “SecureID” used DES in some custom mode: vague suspicions might be attackable
- Now there are standards for Time and Counter based security, in “Google Authenticator” and similar.
- Some tags are “Time Bombed”: tags die after 3 years, whether battery is OK or not
  - Probably a good policy, although expensive for users

# Phishing Works

- Various “seeds” (the initial secret) have been stolen, and users have to inherently trust the manufacturer
- SecureID normally used with a PIN simply concatenated with the code: provides little additional security
- Yubikey can be reprogrammed with new secret material, which helps





# Weaker form

- CRAM-MD5, DIGEST-MD5
- Server knows user password **IN PLAIN TEXT**
- Server sends random
- Client hashes random with password and replies
- Server repeats computation and checks knowledge of password

# CRAM-MD5

- Not as good as a token: single, unchanging password
- Secure against an observer without needing complex crypto
- Requires unhashed secrets on server (but so do hardware tokens)
- Popular for IMAP and SMTP as good fit to protocol
  - Make sure email password is not your main password if CRAM-MD5 or DIGEST-MD5 are in use

# Another weaker form

- ssh public key encryption
- ssh user holds private key
- ssh server holds public key
- server sends random, which is signed by client, checked by server (roughly: it's actually more complex than that)
  - Can use “signature only” algorithms like DSS/DSA

# Problems

- Private key held by client on machine of unknown security
- Key is encrypted with passphrase chosen by user, typed by user on their own computer
  - And often stored in an “agent” unencrypted
  - No publicly known breaks, but probably not the best way to secure logins from home/BYOD machines to machines holding high-value data.

# Password Tunnelling

- Sending plaintext passwords over an encrypted channel feels bad
  - Prone to client-side malware
  - But may not be as bad as it looks
  - Risk analysis, risk analysis, risk analysis

# OAUTH

- Mechanism for doing federated authentication: “I will accept you are MrX@SiteY, because Site Y says so”.
- Technology behind “Log in with Google / Twitter / Facebook”.
  - And leverages their two-factor authentication.
  - Focussed on HTTP authentication but can be used for other things
  - Current project to add it to ssh, for example

# OAUTH

- Complex to implement, hard to implement right
- Basic protocol looks sound (I modelled it in ProVerif, although I haven't published it)
- Lots of places to get implementations wrong, and my gut feel is that there are likely to be a lot of “practical” attacks at the interfaces between the components and in the (relatively complex) crypto: ProVerif doesn't model this.
- But currently looks like a good option to examine
- <https://oauth.net>

# Conclusion

- TLS complex, bug-ridden, but only serious contender if you need confidentiality and integrity
- One time passwords with tokens may be enough for some applications: risk assessment required
- Does no harm to use one time passwords over TLS!
- Two factor authentication: always good



# Security 15: DoS, Amplification, DNS Attacks

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Overview

- What are DoS attacks?
- How are they performed?
- How do we stop them?
- How do we stop our sites and apps being involved?
- Attacks on DNS

# What is a DoS attack

- Denial of Service
- Sometimes about killing applications
- Sometimes about killing machines
- Sometimes about killing networks
- Motives are many: malice, amusement, blackmail, competition, politics...

# How are DoS attacks performed

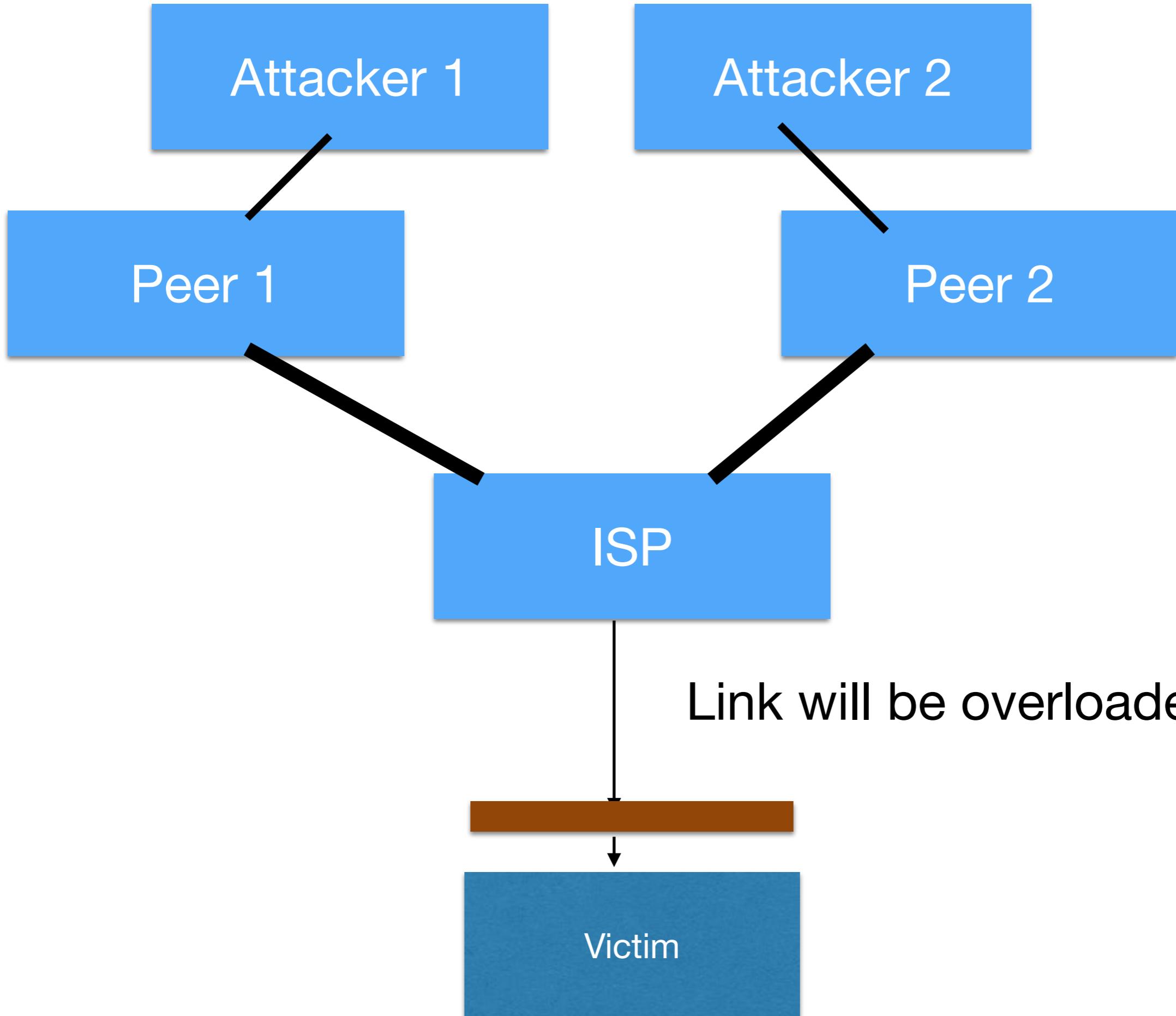
- Crude: lots of packets
- Less crude: lots of packets that do something (SYN, for example)
- Sophisticated: attacks that rely on knowledge of applications

# Crude attacks

- Require one or more of:
  - Fast connection under your control (typical university has  $n \times 10\text{Gb/sec}$  links)
  - Lots of slower connections under your control (botnet)
  - Lots of foot soldiers (LOIC, and other such tools)

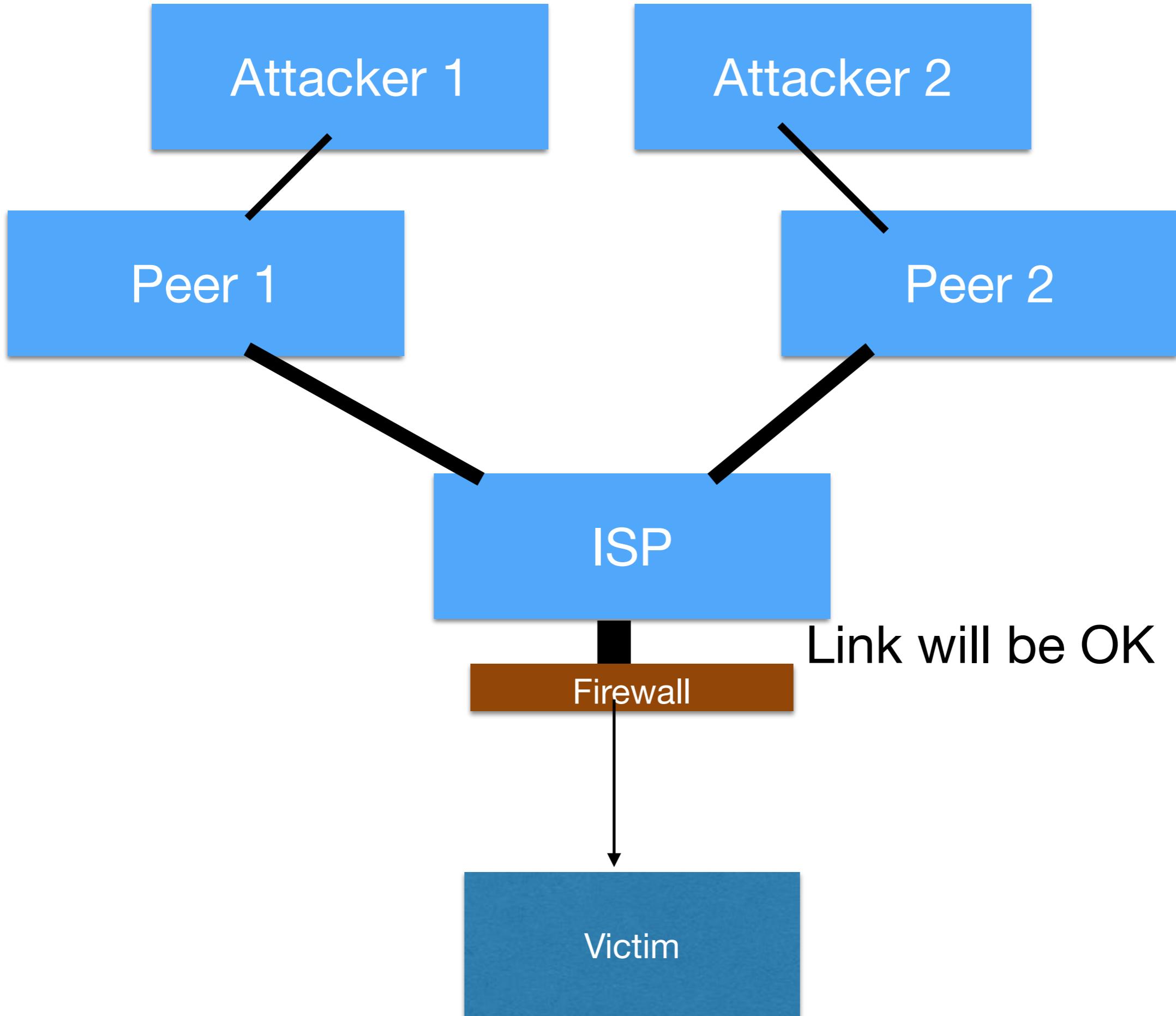
# Crude Attacks

- At crudest, just generate lots of packets directed at victim network
- Victim firewall is ineffective, because the main target is the victim connection



# Mitigation

- Requires support from ISP
- Strong argument in favour of hosted web presence, which makes defence easier (and means link from ISP to your kit is GigE+)
- Also services like Cloudflare, which claim to be able to stand the load and then pass web traffic to your secret address

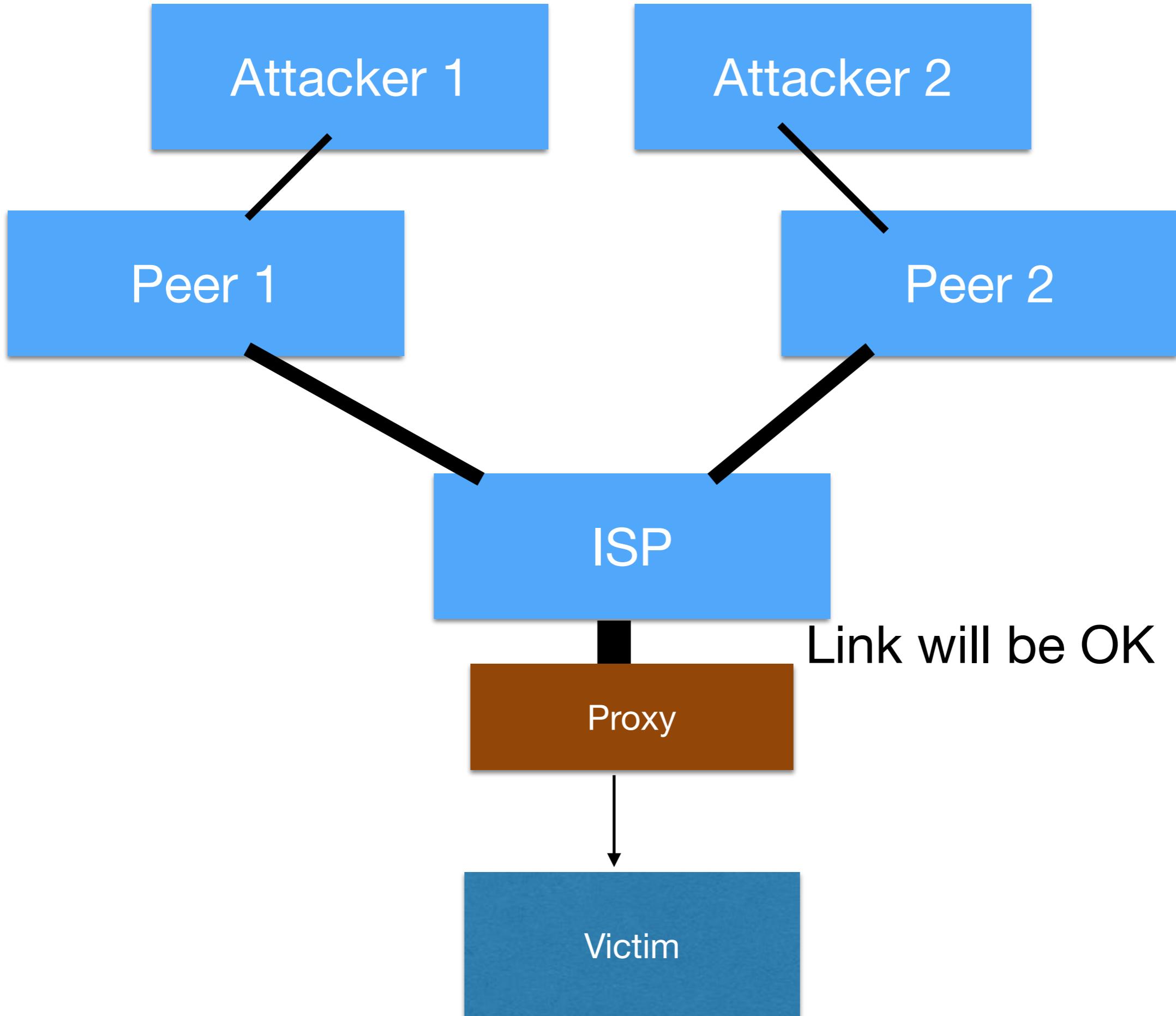


# Mitigation against Idiots

- Let's assume that we are running a web service (ie, port 80 passing http and port 443 passing https)
- Filtering is easy if attacker just spraying packets and random: filter in ISP with simple firewall
- Filtering is almost as easy if SYN flood or similar: filter in ISP with stateful firewall counting unack'd SYN-ACKs.

# But attackers are not idiots

- What do we do if the attacker completes the connection (SYN SYN-ACK ACK) and then starts to send broken / hostile / etc HTTP? Or does nothing, to tie up resources?
- Some solutions involve timeouts (get your content in x seconds or connection is closed anyway)
- Cloudflare: get attacker to run complex Javascript before proxying
  - For naive attackers, their attack scripts can't run the JS
  - For intermediate attackers, their attack script doesn't run the JS quite the same as a real browser
  - For sophisticated attackers, the “proof of work” slows down the attacker



# Encryption?

- Harder to do this if there is https involved
- Requires that intermediary has certificate for target domain
  - That may well breach your security policy, or that of your customer
- Various ad hoc solutions using additional names on intermediary certificate, or extra certificates for the domain (perhaps signed by the intermediary), but governance is tricky, and it complicates pinning.
- And if your content needs encryption, you may be very nervous about a random web company having everything in clear (and your customers may be even more nervous). So contracts and governance are again important.

# CDNs

- For static content, another option is to not serve it from your own kit at all, but serve it from a “content delivery network” assumed to have capacity and capability to resist DoS.
- Examples Akamai and (today) AWS

# Anycasting

- Network 147.188.0.0/16 is Birmingham, and is advertised via BGP by ISP, which propagates it to other ISPs (“peering”)
- All traffic to 147.188.0.0/16 addresses come in via our ISP to our edge router
- There is another way...

# Anycasting

- Have a machine with address 147.188.99.1 in many data centres or peering points around the world
- Advertise 147.188.99.0/24 via BGP in each of those locations
  - BGP requires all netmasks to be  $\leq 24$ .
  - Everyone talks to their local 147.88.99.1, not necessarily the same one
  - Machines also have a unique IP number, in non-anycast block, for management and comms with origin servers. This can be heavily firewalled.

# Anycasting + DoS

- If an attacker takes out one of the 147.188.99.1 copies with a DoS, they will also take out the BGP peering from it
- So everyone fails over to other copies
- Attacker now has to take out multiple core exchanges or big data centres to “win”.

# CDN + Anycasting

- www.bham.ac.uk could point to 147.188.99.1, and serve content from multiple locations
  - Means having lots of (expensive) kit in lots of (expensive) peering points
- Alternatively, www.bham.ac.uk points to bham.some-cdn.com, which in turn uses Anycasting for resilience
  - Pro Tip: Akamai, AWS

# Amplification Attacks

- Examples are DNS and NTP but there are others (all UDP)
- Carefully crafted queries generate large responses, much bigger than the query
- By setting source address of query, third party receives all the responses
- Attacker sends  $n$  bytes to standard applications on  $m$  servers, victim gets  $n.m.a$  bytes, where  $a$  is the amplification factor

# Amplification

- Each use so far has been a bug or misfeature, in that you can fix the problem without breaking the protocol as a whole.
  - For example, debugging interface to NTP that allows you to get details of all your peers in excruciating detail, fixed with patch, firewall and better configuration
- Possible there will be a non-bug attack

# Amplification Mitigation

- This is about avoiding being “that” innocent third party, in advance of knowing which apps have bugs
  - Default Deny
  - Rate limiting
  - Egress Filtering

# App Mitigation

- Don't use UDP!
- If you have to use UDP, don't reply with responses larger than the query, and have strong links between queries and responses (see later DNS problems)

# Egress / Ingress Filtering

- All packets leaving your network should originate in your network; packets arriving over a link should be valid.
- For customer networks, short list (ie, should anything leave the Birmingham network that isn't 147.188.0.0/16?)
- For ISPs, longer list (union of all customer networks) and might be difficult to do, so instead they should use...
- ...ingress filtering on links (ie, should anything come from Birmingham that isn't 147.188.0.0/16)?
  - RFC 2267, January 1998, amplified in RFC 2827, May 2000. **Still not fully, or indeed widely, adopted.**

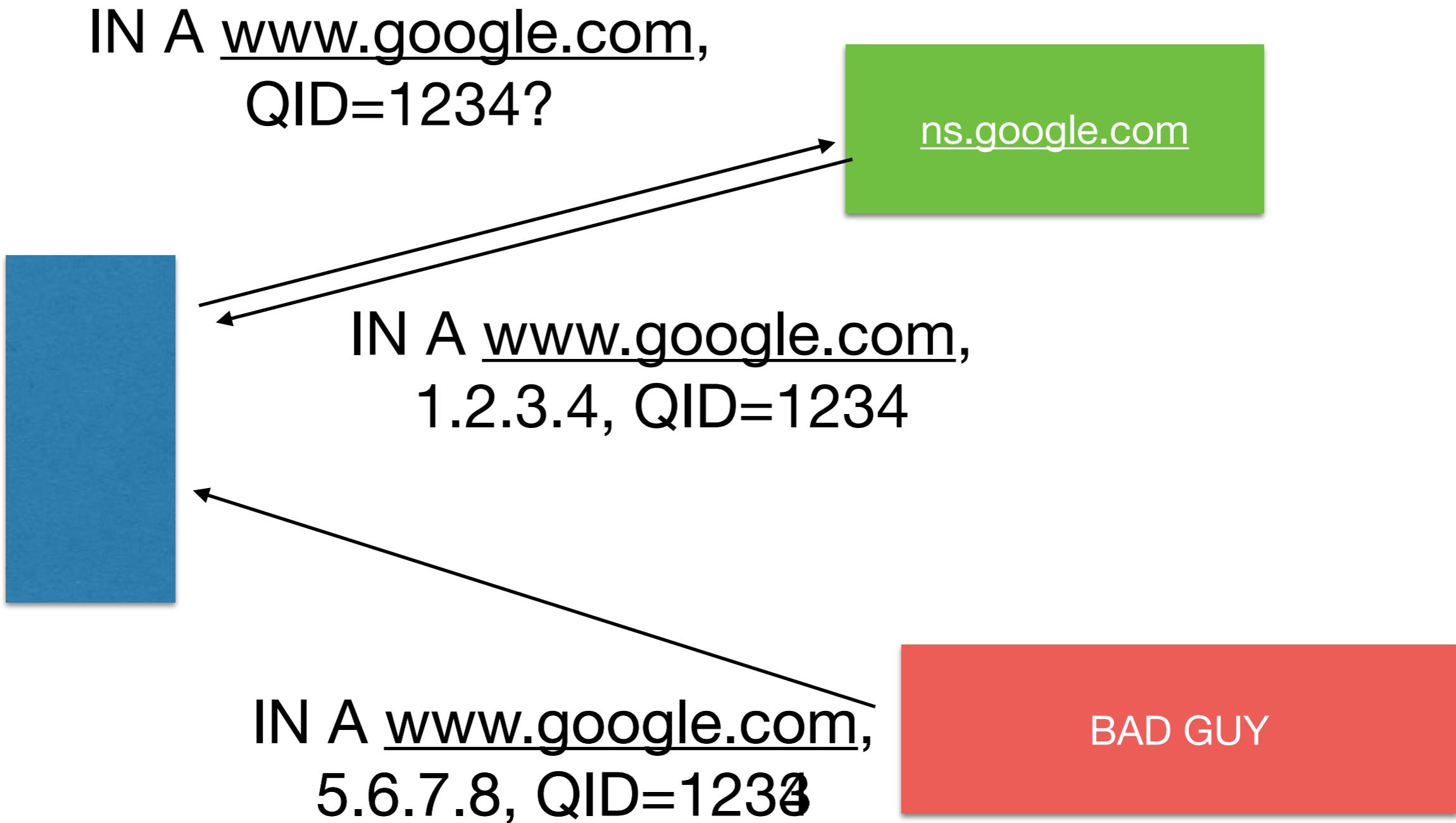
# Egress Filtering

- Sometimes called Bogon or Martian Filtering
- Also worth filtering incoming and outgoing traffic that cannot be right
  - Example RFC1918 (10.0.0.0/8, etc) should never arrive and never leave, and should be dumped in both directions on all firewalls which touch the Internet
  - “How Ian stole email from the US Congress”

# DNS Attacks

- Kaminsky Attack
  - Outgoing DNS requests only have 16 bit QID
  - Bombarding recursive server with responses to “IN A www.google.com? ” cycling over all QID has a non-zero chance to succeed

# Kaminsky Attack



# DNS Attacks

- Mitigation is messy, as fake packet looks exactly like legitimate response to legitimate query
- Problem is that 16 bits means on average only 32k fake responses in order to “win”, which is possible on fast networks
- Caused massive concern (open community 2008, but almost certainly known before then)

# Solution-ish

- Pre 2008, all recursive DNS servers used a single port for outgoing queries
  - Pre 2000 or so, that was port 53, but practice changed because of changes to software
- Post 2008, recursive DNS servers open lots of ports, and randomise queries over them
- 1024 ports = extra 10 bits for attacker to guess if they know which ports are open, extra 16 bits if they don't

# Still only max 32 bits

- Port randomisation does improve matters, and 32 bits makes attack infeasible today
- Not very comfortable safety margin, though, and there will need to be a better fix
- However, **port randomisation lost by passage through NAT layer**
- **Never** run a recursive DNS server behind a NAT point, it is very insecure. **Never. Ever. Ever.**
  - Run it on edge router, if necessary, but better to use ISP server

# Cache Poisoning

- DNS responses are accompanied by “additional information”
  - So answer to “IN MX bham.ac.uk?” might be all the MX records, plus additional information of the IP numbers of those mail exchangers
  - Idea of additional information is to provide stuff you have on hand in your cache

# Cache Poisoning

- Until recently, recursive servers would cache all additional information
- So response to “IN A www.obsuredom.com? ” might include additional information “IN A www.google.com 6.7.8.9”, where 6.7.8.9 was under an attacker’s control
- Modern recursive servers are aware of this (“out of bailiwick controls”) but risk is still there: the code is a forest of subtle heuristics
- Very broken DNS implementations will accept additional information for your local domain, so attacker can override authoritative records

# Cache Poisoning

- Extra fun if the poisoning is done against NS records
- Best practice: never run recursive name server and authoritative name server in same instance, even if you think access control lists, “views” and so on reduce risk. If you need to run an authoritative server, you can afford an extra IP number and an extra VM (or sub-contract it out)
  - Use a “stealth master” and a commercial secondary service

# Human Factors

- Also possible to take over the registration and just change the NS records
- Vital that this does not happen: proper protocols between registrar and domain owner

# Summary

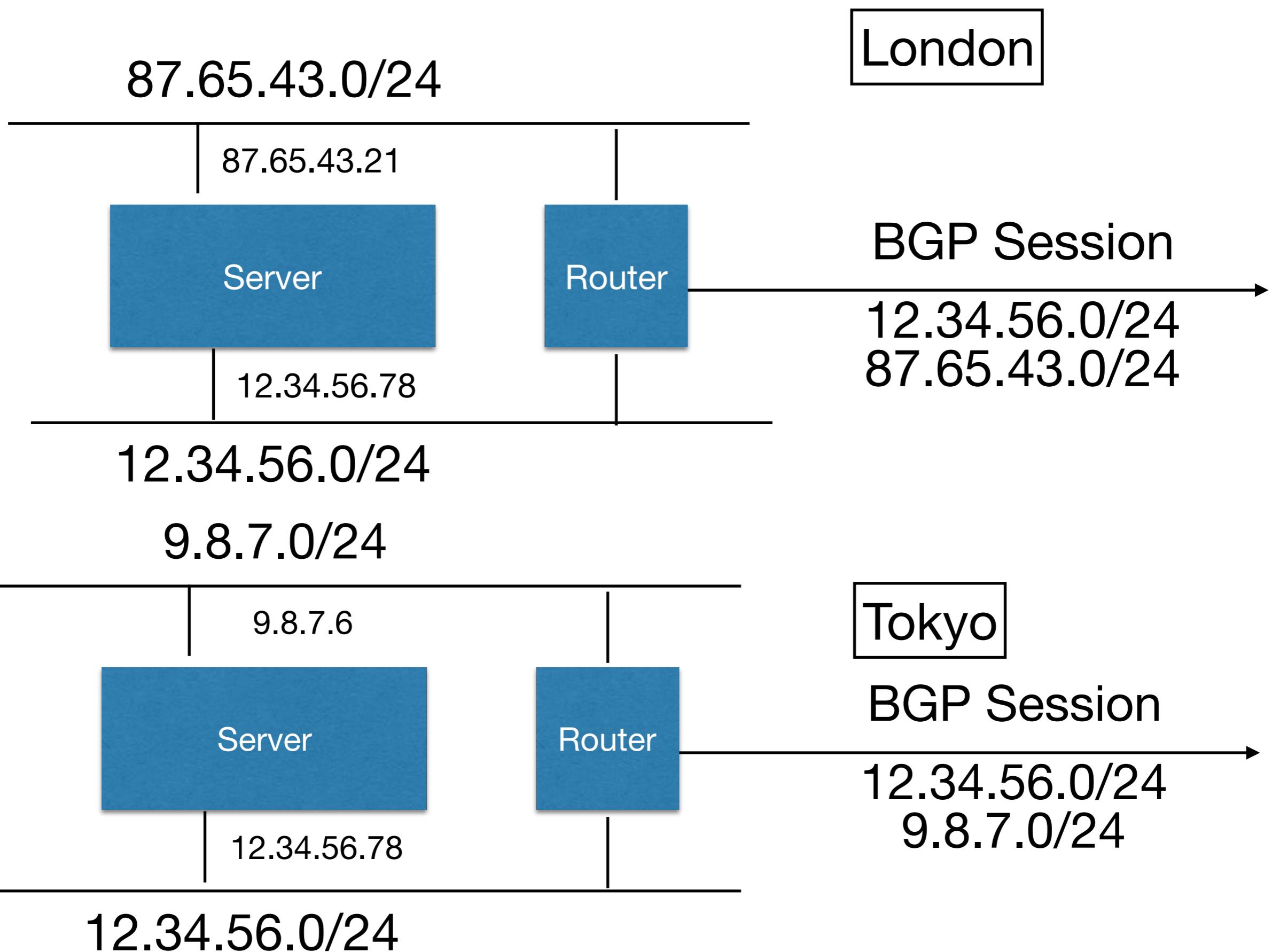
- Egress Filtering
- Default Deny
- Protect and segregate DNS servers by role
- Don't NAT DNS

# Security 16: Anycasting Redux, VPNs

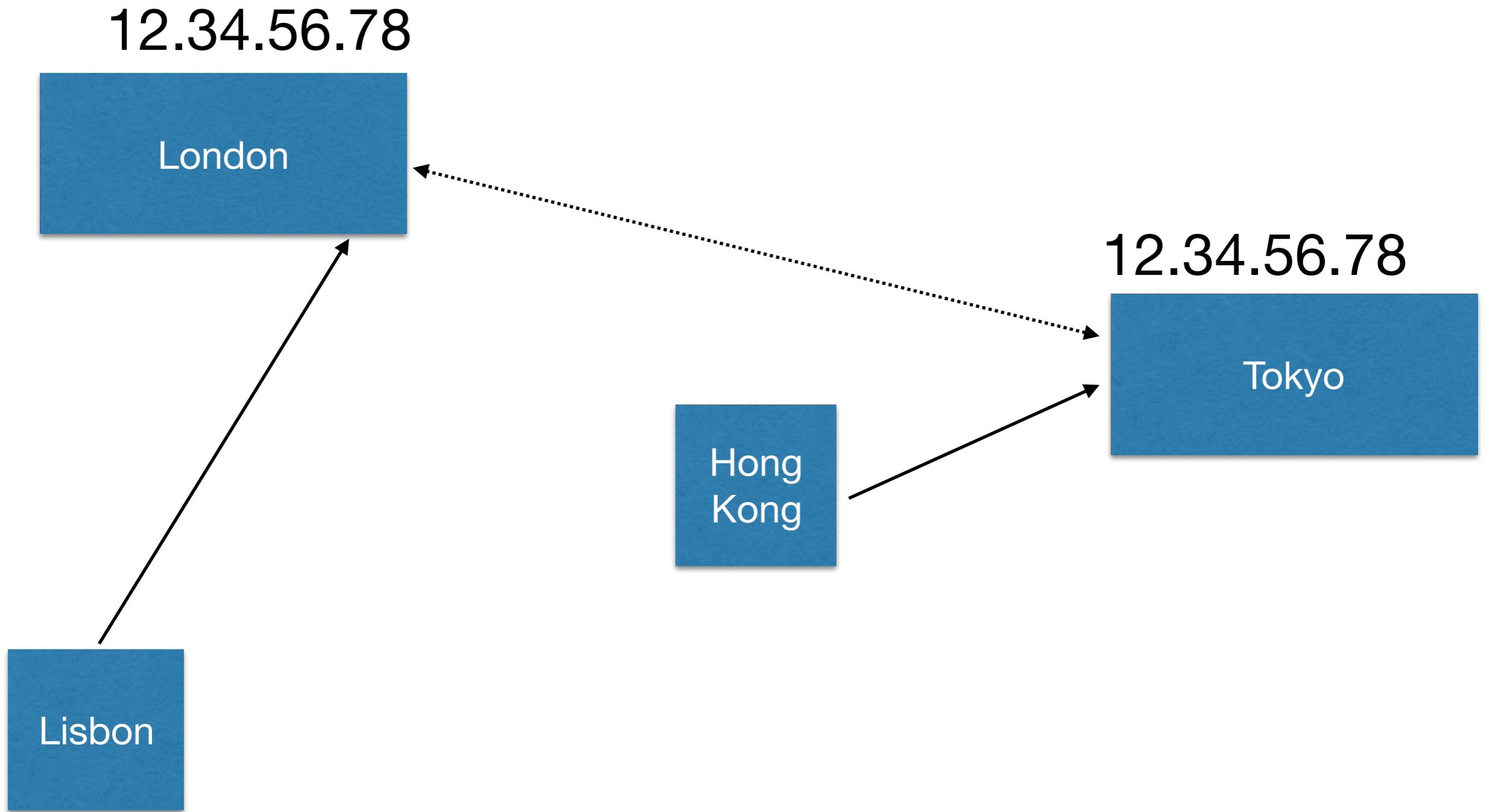
[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Worth Reading (esp. MSc Cyber Security Students)

- <https://www.ncsc.gov.uk/blog-post/tls-13-better-individuals-harder-enterprises>
- <https://www.ncsc.gov.uk/node/2252>
- Serious guy (NCSC Technical Director, ex-GCHQ Technical Director) on balancing improved crypto against enterprise issues.
  - I disagree with his conclusions, but it's still very much worth reading.



# Client



# Anycasting

- 12.34.56.78 is “Anycast”: lots of alternative locations (two in this case, London and Tokyo) for the same service (presumably)
- 9.8.7.6 and 87.65.43.21 are standard “Unicast”: uniquely identify a machine.

# Anycasting

- You can replicate data between the instances using their unique IP numbers, and then access it using the shared address.
- Failover is the BGP convergence time, a few minutes worst case
- Alternative is multiple A records, but failover requires removing the machine from the RRSet and waiting for caches to flush (can take up to an hour, as some systems override short TTLs)
  - And Multiple A records doesn't give geographic sorting

# Anycasting

- Best fit to UDP services, as service will survive route instability
- Bad fit to long-lived TCP connections, as routing changes will leave connections in a bad state
- Acceptable fit to short-lived TCP connections, such as for HTTP, so used by CDNs.

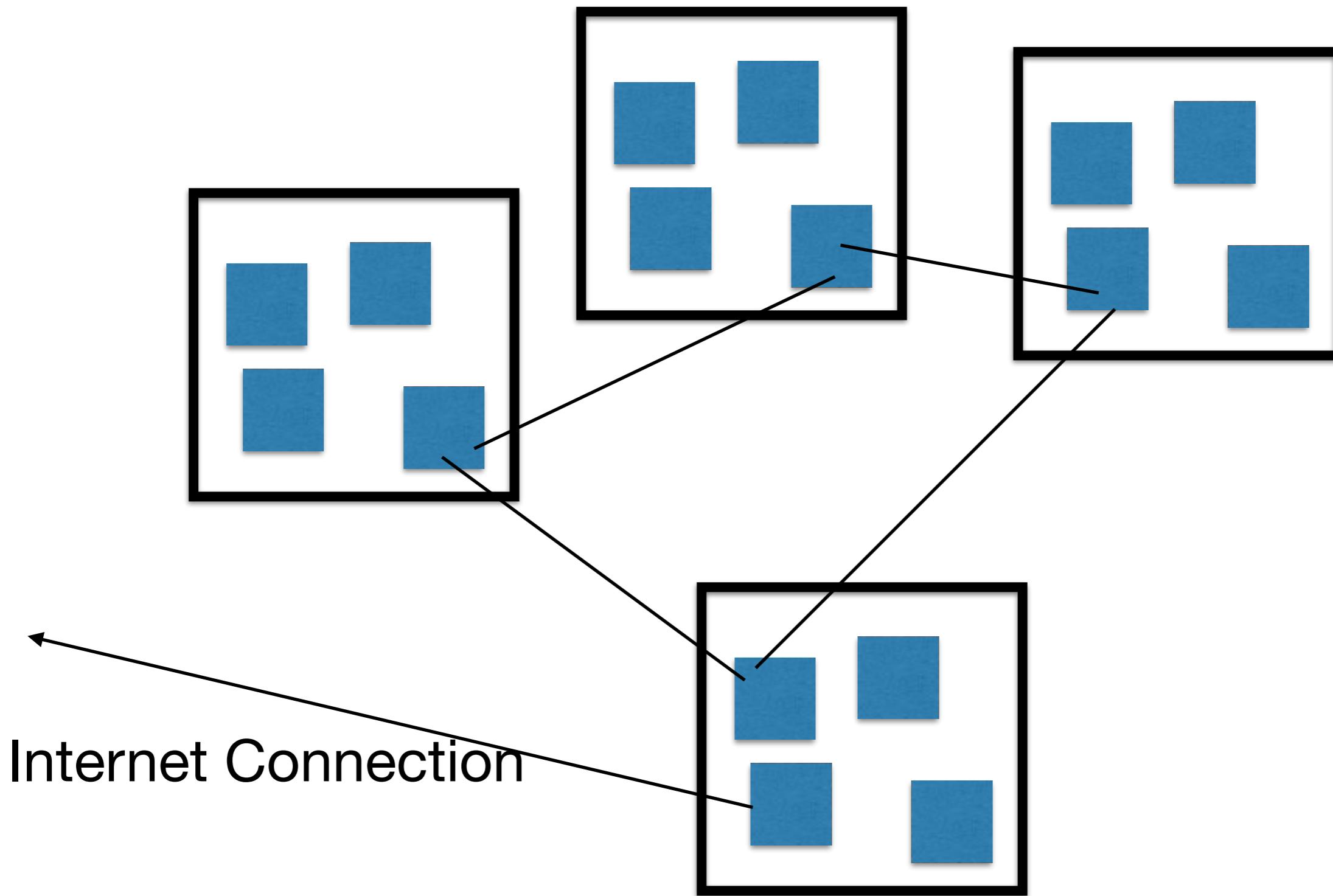
# Database Replication

- If you want to run a database in this sort of architecture, you will probably run the front end application with Anycasting and then commit the data to a redundant pair using some other technology (VRRP, or a proprietary distributed database)
  - Sometimes, the best solution is still a mainframe with “five nines” availability ( $99.999\% = 5 \text{ minutes per year down time}$ ) or “six nines” (30 seconds per year downtime).
  - Running one instance on highly available hardware much, much easier than attempting to do redundancy while maintaining absolute data integrity (banking, airline booking); different if willing to accept inconsistent results during failover (search, shopping).

# VPNs

- Quite a good technology for linked replication servers using Anycasting :-)
- “Virtual Private Network”
- Has come to be linked with remote access for “road warriors” (yes, really) and home working, but is more general than that.

# Multi-Site Company



# Private Networks (not Virtual)

- 1980s/1990s style
- You buy leased lines (kilostream, megastream, E1, DS1,T3: lots of names, lots of speeds, all lots of money) and roll your own network over the top
- Closed user group: as secure as the telco's network

# Dial In

- Remote access 1990s style done with dedicated banks of modems (we had a second phone line at home for my wife to dial into work until about 2002)
- Remote mobile access 1990s/2000s style involved dedicated APNs and the mobile operator delivering data over a leased line to the company
  - If you have to ask the price, you can't afford it

# Internet Changes It

- Internet connectivity much cheaper than international leased lines (although probably not as reliable)
- Reasonable to assume all staff have broadband Internet, but cannot use it to “dial in”: it’s Internet only
  - We tried to put this functionality into the original Project Ascot, but lost
- Mobile internet massively, massively cheaper than previous solutions
- Security issues substantial.
- So we can run it all over the Internet, yes?

# Security Issues

- Running “line of business” applications over the internet extremely risky
  - Not engineered for it, either as protocols or as implementations
- No encryption
- Servers unlikely to survive exposure to outside world for long
  - Address filtering doesn’t work for home users or mobile users

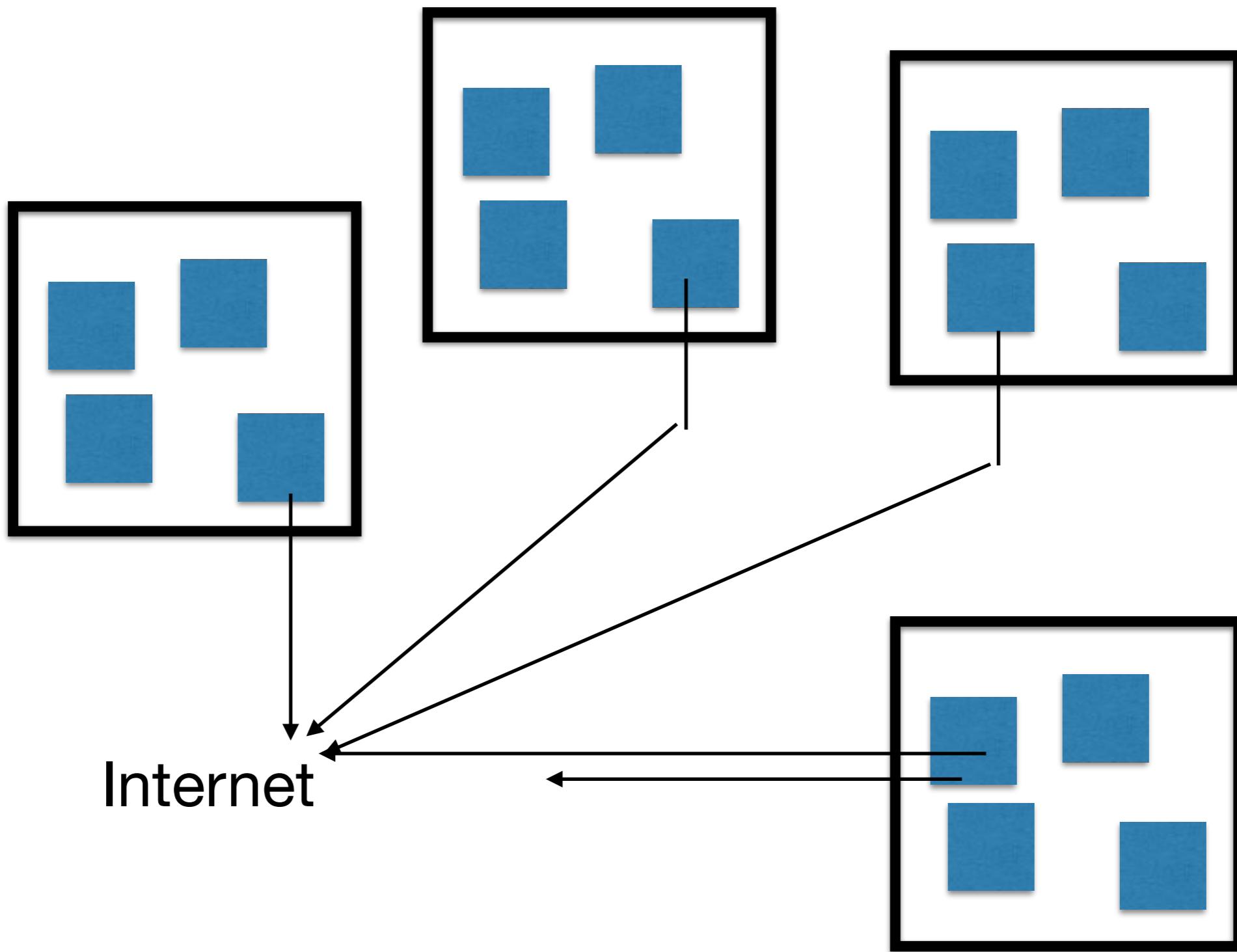
# Networking/OS Issues

- Probably using RFC1918 IP numbers internally, including on LoB services
- May be forced to run old, obsolete, unpatched (or unpatchable) servers for old applications
- May not support any sort of secure authentication

# Hence, VPN

- **Virtual** Private Network
- Tunnel traffic between sites by embedding the packets into IP packets

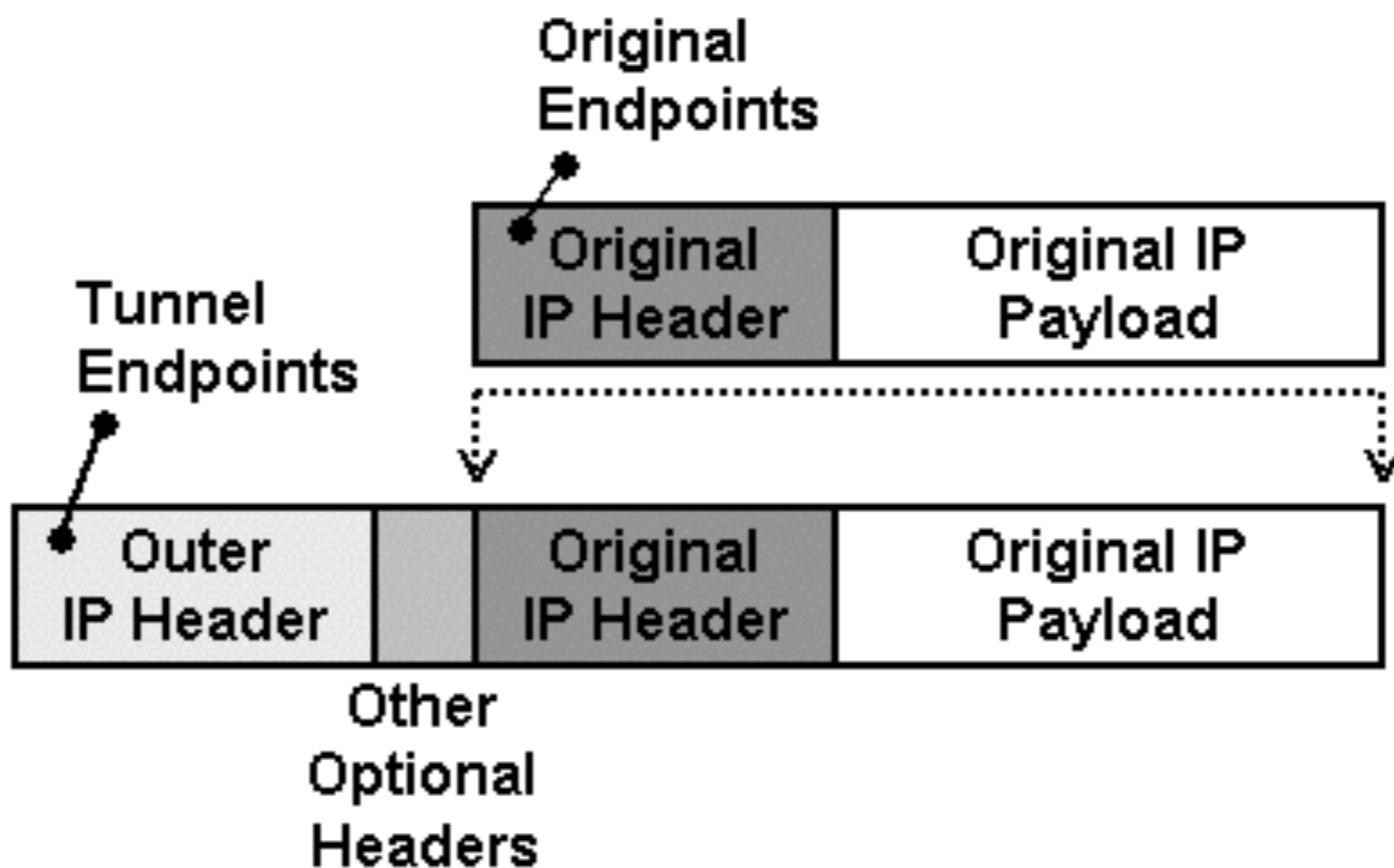
# Multi-Site Company



# IP-in-IP

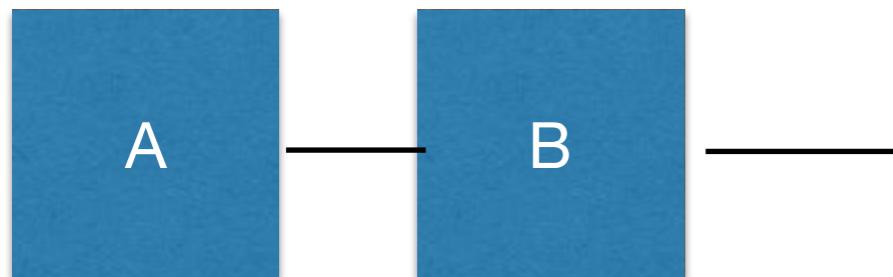
3

## IP-in-IP Encapsulation (1)

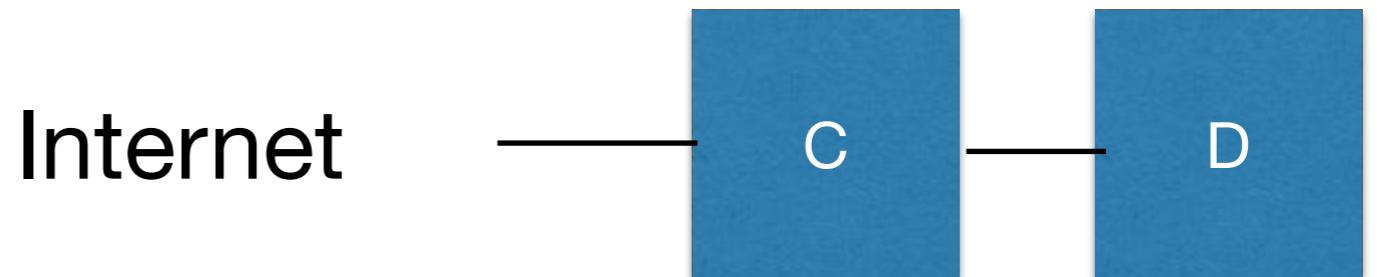


# IP in IP

Site Left

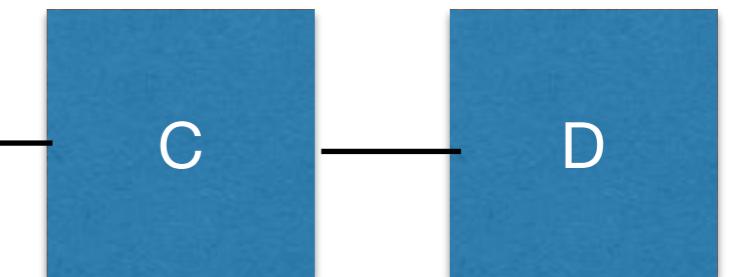


Header: A->D  
*Payload for App*



Header: B->C  
Header: A->D  
*Payload for App*

Site Right



Header: A->D  
*Payload for App*

# Alternatives to IPIP

- GRE (Cisco)
- Tunnelling PPP over TCP (Messy and unnecessary)
- L2TP (Microsoft, Apple)
- All work roughly similarly: a packet and its header become the payload in another packet
  - Can also carry raw ethernet over similar techniques

# Encapsulation

- Edge router of local network has routes to other networks in the VPN, which it seems as being directly attached, one hop away
- Packets from site left to site right are used as the payload of a packet going from router B to router C

# Benefits

- Sites are linked without needing to expose systems to Internet (firewall set to “pass traffic to other end of tunnel, drop everything else”)
- Can have single central point of Internet egress and ingress
  - In practice, web traffic often happens locally
  - RFC1918 is OK so long as the sites don’t overlap
  - But...

# Security?

- In this case, non-existent
- Anyone who knows of the existence of the tunnel can inject packets into it undetectably
- Anyone who can see packets can forge connections undetectably
- If the tunnel medium is TCP it's slightly harder, but still easy enough
  - TCP connection attacks: insert packet at current sequence number, and modify all subsequent packets to include the offset.

# What can we do about it?

- Depends on our attacker model
- We might just need to authenticate packets, so that insertion is impossible
- We might want to encrypt packets for confidentiality
- We probably want to do both

# Technologies

- IPsec (“IP Security”) – last week
  - either directly in tunnel mode (Cisco) or to protect something like L2TP (Apple)
- SSL-VPN: various uses of SSL-type encryption to handle a stream of packets
- Proprietary boxes (after all, so long as it’s reversible and the headers are IP, the payloads can be processed how you like)
  - Crypto.AG and others sell wire-speed AES boxes

# Home / Mobile Use

- Special case where one of the “sites” is a home or mobile user
- Normally we don’t want to link the whole network, just the single machine
  - And arguably, we want to isolate/minimise contact with other machines on the home network

# VPN addresses

- It will usually be difficult to use a VPN to connect to a network using a particular address if you are coming from a machine with the same address.
- Concrete, nasty examples: 192.168.0.0/24, 192.168.1.0/24, 10.0.0.0/8
- For home/SME use, pick either 10.X.Y.0/24 for random X and Y both 1..254, or better (I wish I'd done this) 172.X.Y.0/24, where X 16..31 and Y 1..254.

# Personal VPN

- So single-hop route from local machine to central site
- With **no** IP forwarding on local machine, so only traffic to/from the local machine goes to/from central site
- But...

# Split Tunnelling

- Big debate
- Traffic from employee's laptop can go to company network
- But can company laptop access (for example) printers on home network while connection is up?
- And what about google and so on. Where does the traffic go: via centre, or “direct”?

# Split Tunnelling

- Usually, VPNs are configured to block all traffic to and from the laptop other than from the VPN server, and default route points down VPN connection.
- This is the default on iPhones, for example
- I think arguments are nuanced, but I am in a minority

# Network Security 17: IPsec

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# IPsec

- IP Security
- Provides mechanisms for encrypting and authenticating traffic between hosts and between networks
  - Transformation is packet by packet and operates on any IP packet, so works for UDP, TCP, ICMP, OSPF...
- Developed as part of IPv6: original statement was IPv6 implementations MUST support IPsec
- RFC6434 changes that to SHOULD, because plenty of IPv6 applications don't need / can't use IPsec and requirement for IPsec was (another) perceived barrier to deployment.
- IPsec also back-ported to IPv4 and ships on most IP stacks.

# Structure

- First session: what happens to packets
- Second session: automatic keying
- Third session: a walk through a real (well, my) network, with three different, latest bits implementations, lots of cipher suites, certificates, all mod cons.

# RFC6434

## Delicious understatement

A range of security technologies and approaches proliferate today (e.g., IPsec, Transport Layer Security (TLS), Secure SHell (SSH), etc.) No one approach has emerged as an ideal technology for all needs and environments. **Moreover, IPsec is not viewed as the ideal security technology** in all cases and is unlikely to displace the others.

# IPsec Concepts

- Two modes of encapsulation: tunnel and transport
- Two modes of use: AH (authentication), ESP (encryption) (which can also be combined for bonus complexity)
- Two mechanisms for keying: static and dynamic (IKE).
- Lots of options for cipher suites, key lengths, extra features and chocolate sprinkles
  - Lots of opportunities for implementations or configurations which don't interwork for subtle and difficult to fix reasons.
  - For extra fun, in some cases will agree keys, establish relationship and then pass no traffic as (for example) hash functions are being used slightly differently so all packets fail to verify.

# IPsec Layers

- Data plane transformation of packets has to happen at wire-speed or near-as, so focuses on performance.
- Management layer (key exchange) much less demanding.
- On computers, transformation in kernel or hardware accelerator, key exchange in user-space.
- On routers and other network elements, transformation in hardware or in network processor, key exchange on management card/processor.
- Result: even more complexity, as the cipher suites available are different, as are the implementations (and, if you are doing this seriously, the threat models are different as well).

# IPsec Problems

- Implementation extremely complex
- Lots of options, some conflicting
- Key management issues
- Cynics ask if it was deliberately made too complex to analyse properly

# IPsec Reassurance

- IPsec is accredited for use in classified environments up to the highest, subject to the underlying cipher suites and key management being appropriate and correct management
  - IPsec itself is not the weakest link: simple crypto constructions using well-proven building blocks
- But key exchange is complex and difficult to get right, so **practical** attacks on IPsec probably exploit this
- Manual keying used by the nervous: you can use your fancy spooky RNG to generate 256 bits and use a man with a briefcase, a chain and a gun to move it to the other end. Lots of tamper-resistant/evident hardware needed.

# Cipher Suites

- For IPsec to work, you need (assuming we are encrypting) to agree:
  - A bulk cipher (today AES, historically 3DES, other options available but may not work) in a mode (CBC, CTR for pipelining, GCM to include authentication)
  - A hash function (today ideally SHA256 but often SHA1 owing to poor standardisation, historically MD5 or assorted MAC constructions)
    - Standard truncates SHA256 to 128 bits, but implementations also do 96 and 160, which then don't interwork.
  - AES-CBC(128..256)+SHA1 pretty much guaranteed to work, everything else something of a lottery.

# Cipher Suites

- Automated key exchange also requires a key establishment protocol (DH) plus some means to do mutual authentications (shared secrets, X509 certificates) plus optionally some means to do PFS (DH again)
- Break this and you have access to the keys used for the bulk ciphering.
- Complex, tricky to implement correctly, hard to assure.

# Implementations

- Writing the lower layers (packet processing) fiddly rather than difficult, and intimately tied into the network processing portion of your operating system or hardware.
- Therefore most implementations specific to the OS involved.
  - Two free stacks for Linux, KAME and FreeS/WAN
  - Userland not difficult to write, but hard to get right

# Linux History

- Two projects: KAME (Japanese research consortium) and FreeS/WAN (John Gilmore et al).
- KAME kernel drivers merged into mainline kernel
- FreeS/WAN needed out-of-tree kernel modules
- Later work standardised kernel management interface, so FreeS/WAN userland can use standard kernel (ex-KAME) drivers
- KAME not active, but IKE daemon (“racoon”) used in OS/X. “ipsec-tools” package for most Linuxes available but deprecated.
- FreeS/WAN (dead) forked to OpenS/Wan (almost dead) and StrongS/WAN (main player now); LibreS/WAN is fork of OpenS/Wan with different priorities. IKE daemon pluto (older) or charon (more modern).
- Interworking is painful but possible.

# IPsec modes

- AH (authentication header): Provides integrity of payload and immutable parts of header.
- ESP (encapsulated security payload): Provides encryption and optionally payload integrity.
- AH+ESP: provides full integrity and encryption, but tending to be dropped from modern code in favour of other mechanisms that do the same basic job (StrongS/WAN “AH+ESP bundles are not supported.”)

# AH v ESP

- **AH: Authentication Headers**
  - Keyed hash of the whole packet and the immutable parts of the header
  - Ensures packet has not been changed in transit, and comes where it claims to have come from

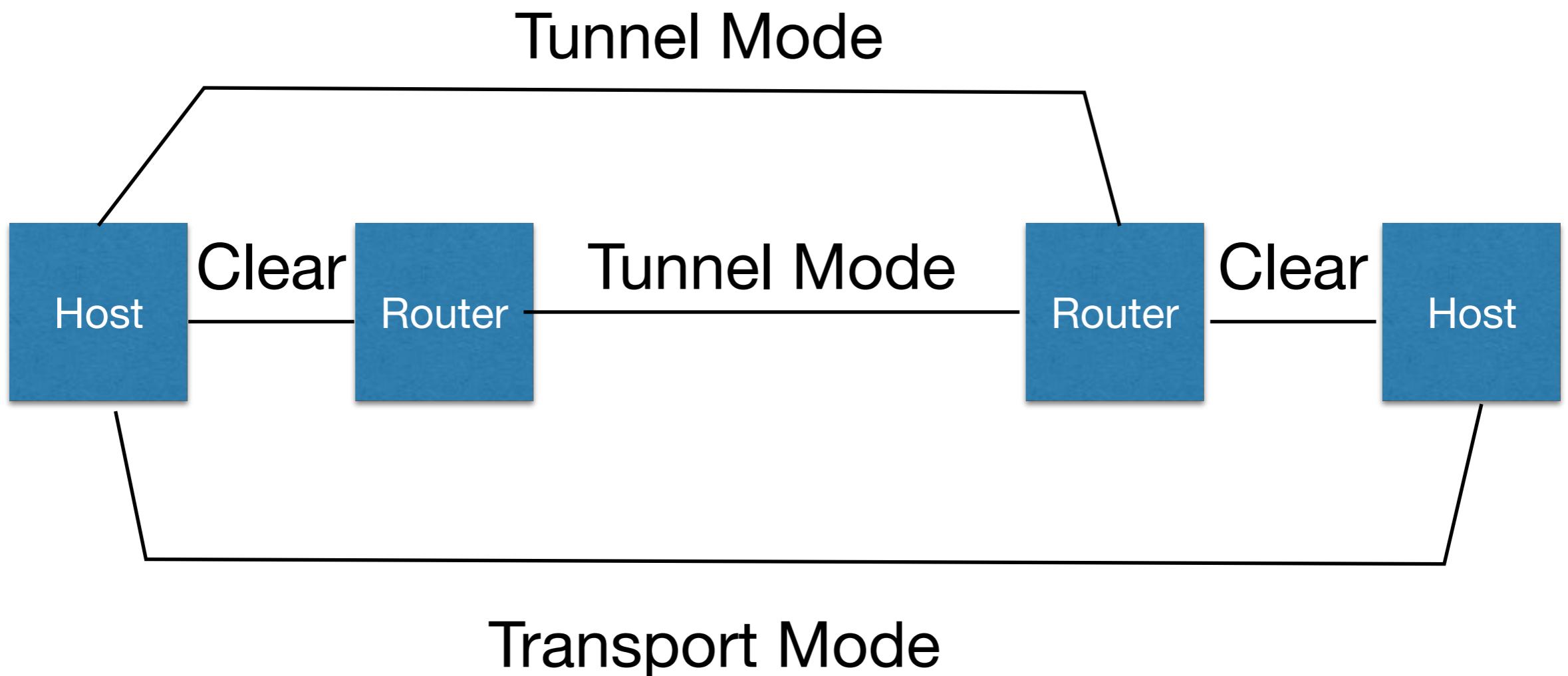
# AH v ESP

- **ESP: Encapsulated Security Payload**
- Encrypts some or all of the packet so that it is confidential
  - Authentication optional, but strongly recommended.
  - Provides Confidentiality, and Integrity if used sensibly

# Transport v Tunnel

- Transport mode is for communication between devices which themselves speak IPsec: packets are transformed directly.
- Tunnel mode is for communication between routers carrying traffic from other end points: packets are encapsulated (IPIP) and then transformed, so IP header is transformed as well.

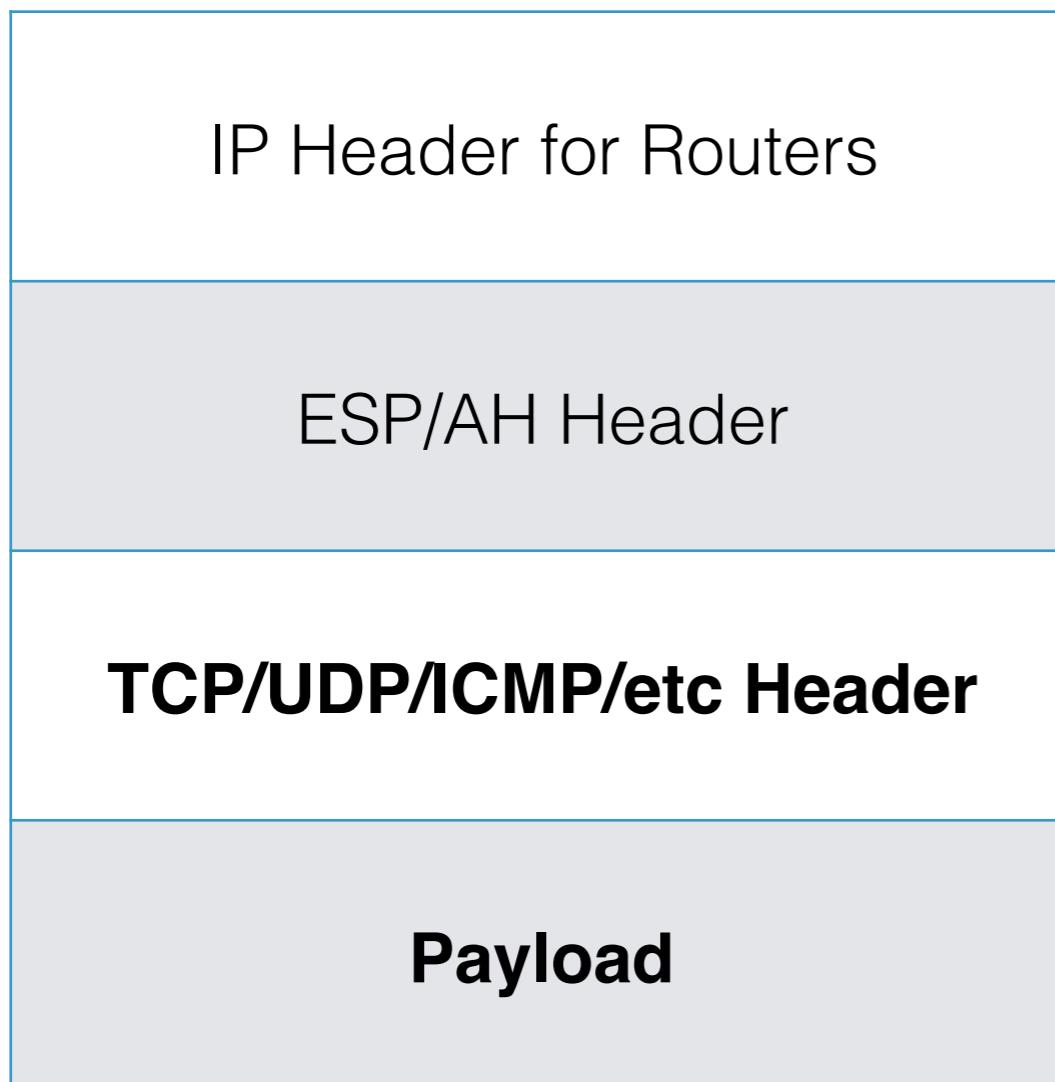
# Transport v Tunnel



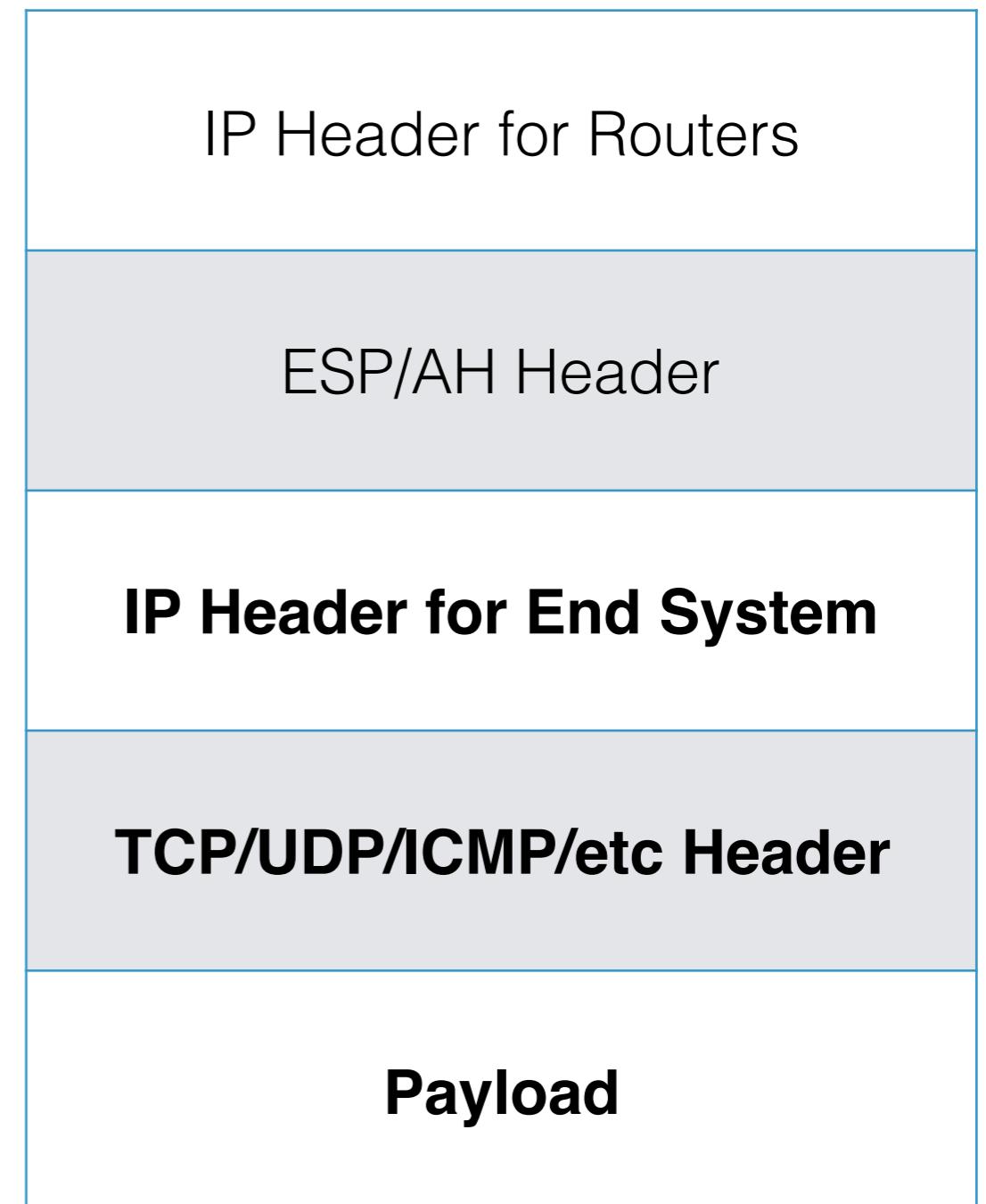


# IPsec structure

## Transport Mode



## Tunnel Mode



# Security Parameters (SPI)

- Defines the **algorithms** and **keys** in use for a **security association**
- SPI is (usually) chosen at random: chance of collision assumed to be vanishingly small
  - You can set them manually in some implementation: choose wrong just doesn't work.
- Each packet is encrypted, decrypted, signed or verified by looking up the SPI in a table

# Security Parameters

```
src 2a04:92c7:e:5db::965d dst 2a00:1630:66:25:f585:a218:7189:22b
proto esp spi 0xc49393c8 reqid 518 mode tunnel
replay-window 32 flag af-unspec
auth-trunc hmac(sha1) 0x16e9338ae105dab513c250a8b281948413dc5728 96
enc cbc(aes) 0x18436568ebf6af3d02fd70ab70485149
src 2001:8b0:1111:1111:0:ffff:5102:4fdc dst 2a04:92c7:e:5db::965d
proto esp spi 0xc6137585 reqid 521 mode tunnel
replay-window 32 flag af-unspec
auth-trunc hmac(sha256) 0xf67e61cbc06...lots chopped...15b57 128
enc cbc(aes) 0xfb4e11a8c8c952b3db136eae1c8db25
src 2a04:92c7:e:5db::965d dst 2001:8b0:129f:a90e:3141:5926:5359:3
proto esp spi 0xc88763fa reqid 517 mode tunnel
replay-window 32 flag af-unspec
aead rfc4106(gcm(aes)) 0x16ef97036f5c64b06644b1a6be2afdd9f7ec1996 128
```

- Three SPIs, AES+SHA1, AES+SHA256, AES-GCM.
- Note different truncations (96, 128), and that AES-GCM doesn't need separate authentication.
- Note I have chopped section out of the (much larger) SHA256 secret

# AH (protocol 51)

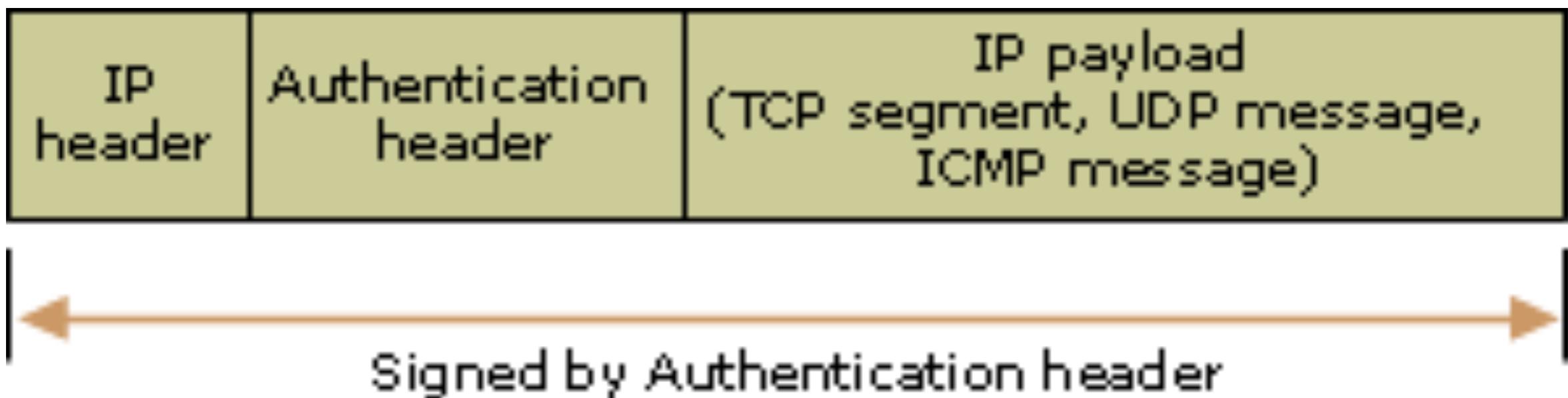
# AH Headers

- Next header: protocol number of next header (usually indicating 17 UDP, 6 TCP or 1 ICMP)
- SPI: Security Parameters Index. 32 bit quantity identifying this particular IPsec relationship.
  - **NOT** the key, but a pointer to the key

# ICV

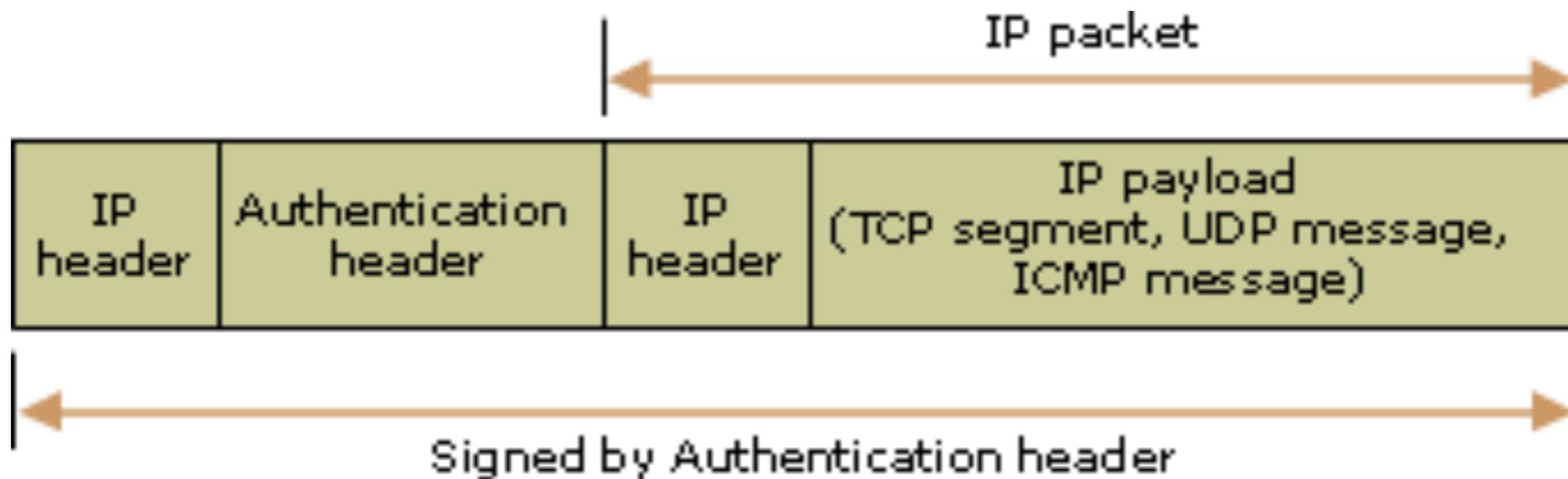
- Integrity Check Value
- A Keyed Hash, Keyed MAC or some other object that provides the assurance you need
  - Agreed “out of band” by participants for manual keying, or as part of IKE negotiation for automatic
- Causes interworking problems as truncation of hash functions often got wrong, and then other implementations need to interwork with broken code.
  - Matrix of interoperable pairs worryingly sparse for anything other than SHA1.

# AH Transport



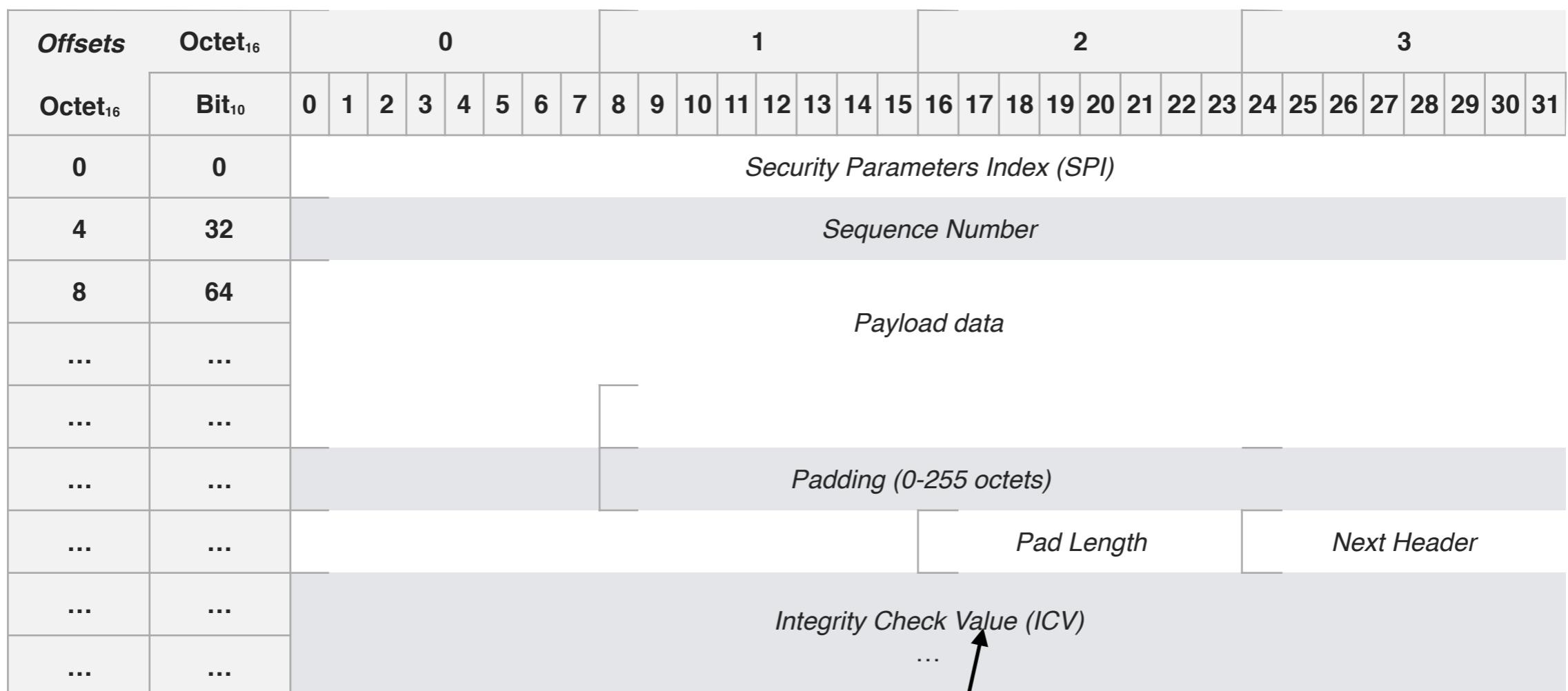
IP Header is proto 51,  
“Next Header” is proto 6 for TCP,  
Payload is then a TCP header and data

# AH Tunnel



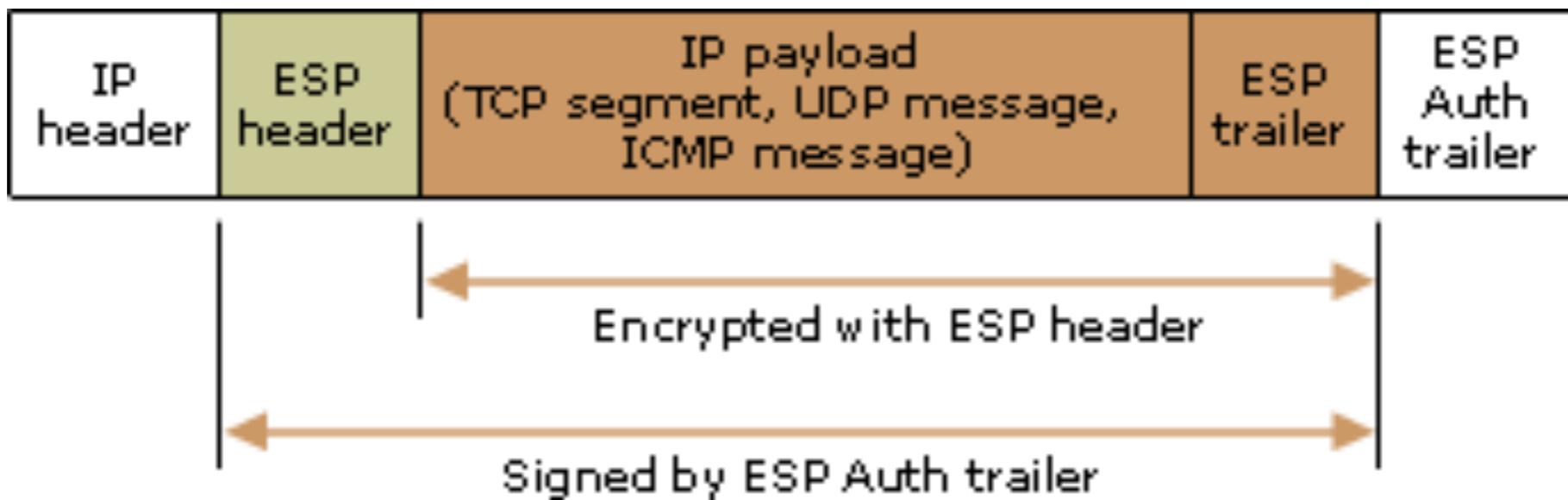
Outer IP Header is proto 51,  
“Next Header” is proto 4 for IP,  
Inner IP Header is proto 6 for TCP  
Payload is then a TCP segment

# ESP (Protocol 50)



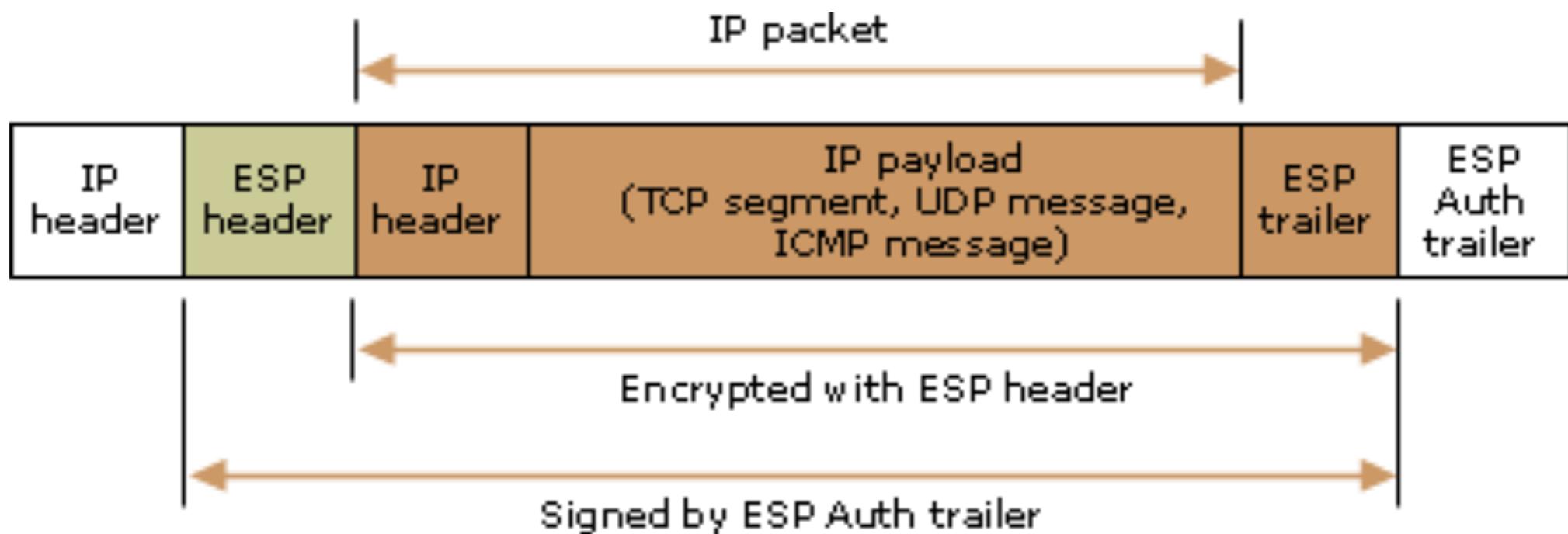
Optional “Authentication Trailer”

# ESP Transport



IP Header is proto 50,  
“Next Header” is proto 6 for TCP,  
Payload is then a TCP header and data  
Auth Trailer is OPTIONAL (but in reality  
it should always be used)

# ESP Tunnel



Outer IP Header is proto 50,  
“Next Header” is proto 4 for IP,  
Inner IP Header is proto 6 for TCP  
Payload is then a TCP segment

# ESP does not protect IP header

- ESP in transport mode does not protect the IP header from alteration; in particular, the source IP address can be changed undetectably.
- One choice is ESP in tunnel mode
- Alternatively, use AH and ESP at the same time, as AH includes the immutable parts of the IP header in the signature.
  - This is often not needed, as ESP provides origin assurance if you choose the keys properly.
  - No longer supported in many implementations

# ESP Auth Performance

- Common cipher suites (AES+SHA1, AES+SHA256, AES in CBC or CTR mode) need two passes over the packet, one to make cipher text, one to form keyed hash for ICV.
- CTR and GCM mode popular in hardware as it means keystream can be formed in advance, and only an XOR is needed at the point of transformation.
  - CTR requires an XOR, GCM and XOR and a Galois Field multiplication.
  - Both allow the AES encryption to be done in advance, as part of a pipeline.

# Sequence Numbers

- Can be used to discard old packets to prevent replay attacks, with a window based on risk tolerance and network reliability
- Sequence numbers **not** used to drive any sort of ACK or retransmission strategy, just to prevent replays.

# NAT Problems

- NAT essentially uses port numbers to extend the range of available source or destination addresses
- Port numbers are a UDP/TCP thing; there are no port numbers in IP packets without layering another protocol.
- IPsec is IP, not UDP/TCP, therefore doesn't have port numbers available to do this mapping.
- NAT-Traversal IPsec embeds IPsec into UDP Port 4500 packets (necessary for roadwarrior VPNs, coming soon, as you're sure to be NAT'd by your hotel/mobile operator and probably home). Details in **RFC3947**: be brave!
- Had to be bolted on to protocol later as IPsec was originally for IPv6, remember?

# ESP in Action

```
[igb@offsite6 ~]$ ntpq -p -6 pi-one.batten.eu.org
      remote          refid      st t when poll reach    delay    offset  jitter
=====
xGPS_NMEA(0)    .GPS.        3 l   -  16 377    0.000    0.000  0.004
oPPS(0)          .PPS.        0 l   -   8 377    0.000    0.000  0.004
*ntp0.linx.net  .PPS.        1 u   11  64 377  13.795    0.497  3.041
+ntp1.linx.net  .PPS.        1 u   -  64 377  13.784    0.575  2.221
+ntp2.linx.net  .GPS.        1 u   -  64 377  15.477    0.981  4.928
[igb@offsite6 ~]$
```

# Traffic Flows

```
14:05:52.880509 IP6 offsite6.batten.eu.org > pi-one-out.home.batten.eu.org: ESP(spi=0x054e459a,seq=0x31), length 68
14:05:52.881668 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org: ESP(spi=0x0a632b19,seq=0x59), length 84
14:05:52.925064 IP6 offsite6.batten.eu.org > pi-one-out.home.batten.eu.org: ESP(spi=0x054e459a,seq=0x32), length 68
14:05:52.926505 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org: ESP(spi=0x0a632b19,seq=0x5a), length 532
14:05:52.927140 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org: ESP(spi=0x0a632b19,seq=0x5b), length 116
14:05:52.957383 IP6 offsite6.batten.eu.org > pi-one-out.home.batten.eu.org: ESP(spi=0x054e459a,seq=0x33), length 68
14:05:52.958816 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org: ESP(spi=0x0a632b19,seq=0x5c), length 532
14:05:52.959440 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org: ESP(spi=0x0a632b19,seq=0x5d), length 116
14:05:52.990874 IP6 offsite6.batten.eu.org > pi-one-out.home.batten.eu.org: ESP(spi=0x054e459a,seq=0x34), length 68
14:05:52.992432 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org: ESP(spi=0x0a632b19,seq=0x5e), length 532
```

# ESP Setup

```
spdadd -6 ::/0[53] ::/0[any] udp -P out none;
spdadd -6 ::/0[53] ::/0[any] tcp -P out none;
spdadd -6 ::/0[any] ::/0[53] udp -P in none;
spdadd -6 ::/0[any] ::/0[53] tcp -P in none;

# or ssh
spdadd -6 ::/0[22] ::/0[any] tcp -P out none;
spdadd -6 ::/0[any] ::/0[22] tcp -P in none;

spdadd -6 offsite6.batten.eu.org pi-one-in.home.batten.eu.org any -P out IPsec esp/transport//require;
spdadd -6 pi-one-in.home.batten.eu.org offsite6.batten.eu.org any -P in IPsec esp/transport//require;
spdadd -6 offsite6.batten.eu.org pi-one-out.home.batten.eu.org any -P out IPsec esp/transport//require;
spdadd -6 pi-one-out.home.batten.eu.org offsite6.batten.eu.org any -P in IPsec esp/transport//require;
```

# Keys are configured (how this is done is later)

```
2001:8b0:129f:a90e:3141:5926:5359:1 2a00:7b80:3019:12::579c:4928
esp mode=transport spi=174271257(0x0a632b19) reqid=0(0x00000000)
E: aes-cbc acbfc369 623a8cd4 50f5eeda 42c8be7e 3e3b40fc 6ccc5ece
A: hmac-sha256 07e62293 80e975f8 d52539ac b096fbe1 e5bac932 59989e78 d6f4bdcc ad04d3af
seq=0x00000000 replay=4 flags=0x00000000 state=mature
created: Mar 10 14:03:17 2015 current: Mar 10 14:07:23 2015
diff: 246(s) hard: 86400(s) soft: 69120(s)
last: Mar 10 14:03:28 2015 hard: 0(s) soft: 0(s)
current: 26144(bytes) hard: 0(bytes)soft: 0(bytes)
allocated: 99 hard: 0 soft: 0
sadb_seq=1 pid=15547 refcnt=0

2a00:7b80:3019:12::579c:4928 2001:8b0:129f:a90e:3141:5926:5359:1
esp mode=transport spi=89015706(0x054e459a) reqid=0(0x00000000)
E: aes-cbc 8c041c0f cdae4c14 bb46f4f8 a3700ebf 13e4a50f 6df95ab1
A: hmac-sha256 93f8e3ee 72228356 acf21816 f4e248f4 b6fbf69d e6edf786 90db00ef c7cd54ff
seq=0x00000000 replay=4 flags=0x00000000 state=mature
created: Mar 10 14:03:17 2015 current: Mar 10 14:07:23 2015
diff: 246(s) hard: 86400(s) soft: 69120(s)
last: Mar 10 14:03:28 2015 hard: 0(s) soft: 0(s)
current: 1080(bytes) hard: 0(bytes)soft: 0(bytes)
allocated: 54 hard: 0 soft: 0
sadb_seq=2 pid=15547 refcnt=0
```

# Simple Connection (no IPsec)

```
mini-server:~ igb$ telnet pi-one 1234
Trying 2001:8b0:129f:a90e:3141:5926:5359:1...
Connected to pi-one.batten.eu.org.
Escape character is '^]'.
pi-one.home.batten.eu.org, DNS, DHCP, NTP (GPS), heyu, odds and ends.
```

```
*** This is a private machine. If you are not personally authorised by
*** igb@batten.eu.org then your access is illegal.
```

```
Connection closed by foreign host.
mini-server:~ igb$
```

# Plaintext Traffic

```
14:27:32.822472 IP pi-one.batten.eu.org.search-agent > mini-server.home.batten.eu.org.60350: Flags [P.],  
seq 1:196, ack 1, win 453, options [nop,nop,TS val 14735272 ecr 344971307], length 195  
0x0000: 0026 bb60 07ce b827 ebe1 9651 0800 4500 .&.`....'...Q..E.  
0x0010: 00f7 bcc4 4000 4006 b4a0 51bb 96d3 0a5c ....@.@@...Q....\r  
0x0020: d5b1 04d2 ebbe 42d6 6b62 3361 f57f 8018 .....B.kb3a....  
0x0030: 01c5 f432 0000 0101 080a 00e0 d7a8 148f ...2.....  
0x0040: d82b 7069 2d6f 6e65 2e68 6f6d 652e 6261 .+pi-one.home.ba  
0x0050: 7474 656e 2e65 752e 6f72 672c 2044 4e53 tten.eu.org,.DNS  
0x0060: 2c20 4448 4350 2c20 4e54 5020 2847 5053 ,.DHCP,.NTP.(GPS  
0x0070: 292c 2068 6579 752c 206f 6464 7320 616e ),.heyu,.odds.an  
0x0080: 6420 656e 6473 2e0a 0a2a 2a2a 2054 6869 d.ends...***.Thi  
0x0090: 7320 6973 2061 2070 7269 7661 7465 206d s.is.a.private.m  
0x00a0: 6163 6869 6e65 2e20 2049 6620 796f 7520 achine...If.you.  
0x00b0: 6172 6520 6e6f 7420 7065 7273 6f6e 616c are.not.personal  
0x00c0: 6c79 2061 7574 686f 7269 7365 6420 6279 ly.authorised.by  
0x00d0: 0a2a 2a2a 2069 6762 4062 6174 7465 6e2e .***.igb@batten.  
0x00e0: 6575 2e6f 7267 2074 6865 6e20 796f 7572 eu.org.then.your  
0x00f0: 2061 6363 6573 7320 6973 2069 6c6c 6567 .access.is.illeg  
0x0100: 616c 2e0a 0a al...
```

# Via IPsec ESP

```
[igb@offsite6 ~]$ telnet 2001:8b0:129f:a90e:3141:5926:5359:1 1234
Trying 2001:8b0:129f:a90e:3141:5926:5359:1...
Connected to 2001:8b0:129f:a90e:3141:5926:5359:1.
Escape character is '^]'.
pi-one.home.batten.eu.org, DNS, DHCP, NTP (GPS), heyu, odds and ends.
```

```
*** This is a private machine. If you are not personally authorised by
*** igb@batten.eu.org then your access is illegal.
```

```
Connection closed by foreign host.
[igb@offsite6 ~]$
```

# ESP in Action

```
14:41:13.207506 IP6 pi-one-out.home.batten.eu.org > offsite6.batten.eu.org:  
ESP(spi=0x0a632b19,seq=0x9b), length 276  
0x0000: 6000 0000 0114 3240 2001 08b0 129f a90e `.....2@.....  
0x0010: 3141 5926 5359 0001 2a00 7b80 3019 0012 1AY&SY..*.{.0...  
0x0020: 0000 0000 579c 4928 0a63 2b19 0000 009b ....W.I(..c+....  
0x0030: b035 8270 e510 ffed fcde a419 589b 287c .5.p.....X.(|  
0x0040: 3444 0fd7 9a8a bb6c 7051 a47b 5253 ebbd 4D.....lpQ.{RS..  
0x0050: 7cad d27a 440f 189a afcb f700 8f05 f677 |..zD.....w  
0x0060: 9ab1 da2b d833 f1e0 aed5 bf2f 65e6 d605 ...+.3...../e...  
0x0070: 1f1b 8598 2702 9d07 246e 57f8 c751 3ba3 ....'...$nW..Q;.  
0x0080: 48f3 ba8d 34d8 1ebd 29c7 2fa5 78ef d754 H...4...)./.x..T  
0x0090: 73fb 0413 e029 9c9f 9ca8 2138 557b 4735 s....)....!8U{G5  
0x00a0: da39 9db2 7a8d 0e0e f4aa f024 6977 b1b8 .9..z.....$iw..  
0x00b0: 8fb7 dfcc 840d c740 988c fa0f d30a 1b08 .....@.....  
0x00c0: f505 c00c 8041 52a2 2727 9916 2955 02d5 .....AR.''.)U..  
0x00d0: 4ef2 95dd 118e 2804 9482 9f0a 1b57 269e N.....(.....W&.  
0x00e0: 2277 eb0d 370c 2035 d78d 6071 8aae b11e "w..7..5..`q...  
0x00f0: ea57 8af8 b53a 3cae 752a 91f9 1e1d 7d8d .W....<.u*....}.  
0x0100: 00b6 a304 c851 568b f452 e7e5 bc0b efd9 .....QV..R.....  
0x0110: 708d 02d8 5607 bca7 3c7c c417 3333 5944 p...V...<|..33YD  
0x0120: d7cc e4d0 34ef 2232 5af6 4ffe 401b 77df ....4."2Z.0.@.w.  
0x0130: 02f9 6e10 0aae e5f8 37cb 4518 ..n.....7.E.
```

# Security 19: IKE

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Key Management

- IPsec allows us to use keys agreed between parties, identified by SPI, to encrypt and sign packets.
- Those keys can be manually configured, or automatically set up

# Manual Keying

- Usually done with endpoints and (optionally) a protocol
- “Allocate this flow an SPI, and set up the keying like this”

```
example# ipseckey
ipseckey> add ah spi 0x90125 src me.domain.com dst you.domain.com \
           authalg md5 authkey 1234567890abcdef1234567890abcdef
ipseckey> update ah spi 0x90125 dst you.domain.com hard_bytes \
           16000000
ipseckey> exit
```

# Why manual?

- Simpler
- You control and generate the key material so can use your super-special RNG
- Re-keying and so on is under manual control
- Fewer protocols to assure

# Why not manual?

- Involves generating and conveying key material in a secure manner
- Synchronising re-keying to avoid dropped packets might be a problem (depends on tight clock sync or some in- or out-of-band signalling)
- Hideous as number of nodes rises
- Forward secrecy tricky (although not impossible)

# IKE

- Internet Key Exchange
- Based on Diffie-Hellman exchange
- Designed by people who knew what they were doing
- “Oakley” protocol presumably named after the former GCHQ site in north Cheltenham (cf. Benhall, the southern site that is now the main base)

# Problems

- Key exchange has to take place *ex nihilo*, because the keys are the basis for later secure communication: we can't use IPsec (no keys)
- IPsec developed earlier than TLS, so no ability to use TLS
- Crucially, doesn't assume any sort of PKI, although it can use one if available

# Problems with DH

- Man in the middle
- Assurance of Key derivation functions
- But only game in town for the purposes we have
  - (IKE supports both EC and modular variants, and plugging in other key exchange protocols would be possible)

# Basic IKE flow

- Create an ISAKMP SA (*Internet Security Association Key Management Protocol Security Association*) (“Phase One”)
- Use the ISAKMP SA to protect the negotiation of keys used for traffic (“Phase Two”)
- Very low volumes of traffic over the ISAKMP SA, so ultimate quality of keys has different priorities: a break is VERY serious, but the volumes of ciphertext available to the attacker are small and the protocol protects against chosen/adaptive attacks.

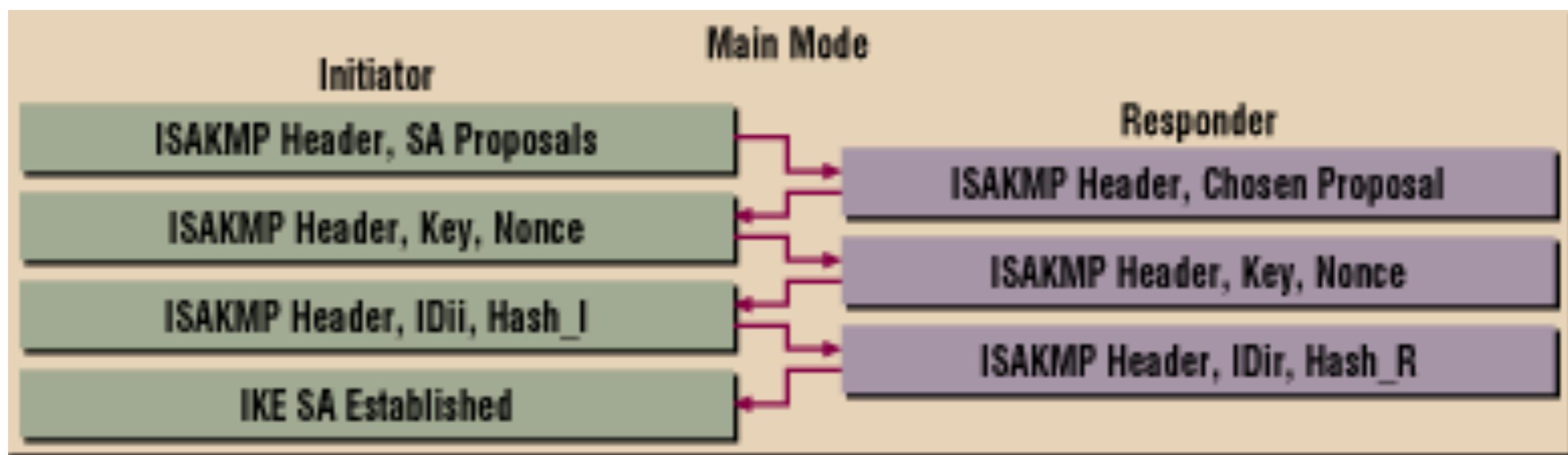
# Authentication Mechanisms

- Pre-shared keys: both parties have a symmetric key that they use to authenticate while building the ISAKMP SA
- RSA Keys: one party has the private, one party the public part of an RSA key pair
- X.509 certificate: signed RSA keys

# IKE Mode: Main

- Slower, involving five exchanges
- Conceals the parties' identities within the protocol

# IKE Main Mode



# SA Proposals

- Which algorithms are going to be used (initiator lists the ones they are willing/able to use, responder sends back the chosen algorithms)
- Nota Bene: these algorithms are protecting the ISAKMP SA, not the eventual flow of IPsec traffic, so slow is OK
  - May be different to final data algorithms because we are in user-space, not in hardware-assist or kernel space

# SA Key Material

- Key material in second exchange is the two sides of a Diffie Hellman exchange, from which a shared key is derived to use to protect the rest of the exchange using the agreed symmetric algorithm

# Authentication

- Both sides then send their identity, and a hash over the whole message calculated using the key material the two parties share (pre-shared key or nonces encoded using RSA)
- Both parties now share a key and are confident about the identity of the other party.
- The hash input also includes the shared key, so any man-in-the middle is detected at this point

# Simple MITM

- Alice sends  $g^a$ , Bob sends  $g^b$ , both can compute  $g^{ab}$
- Mallory can do this exchange separately with Alice and with Bob, so each negotiates a different key. Mallory then decodes traffic from Alice and re-encodes it with the key negotiated with Bob.

# MITM protection

- Alice and Bob share secret S.
- Alice computes  $K = g^{ab}$  and then computes  $h(K \cdot S)$  and sends it to Bob
- Bob does the same thing
- Mallory cannot forge the required packets as Mallory does not know S.
- S only needs to resist attack on hash function, rather than being exposed as a long-term key.

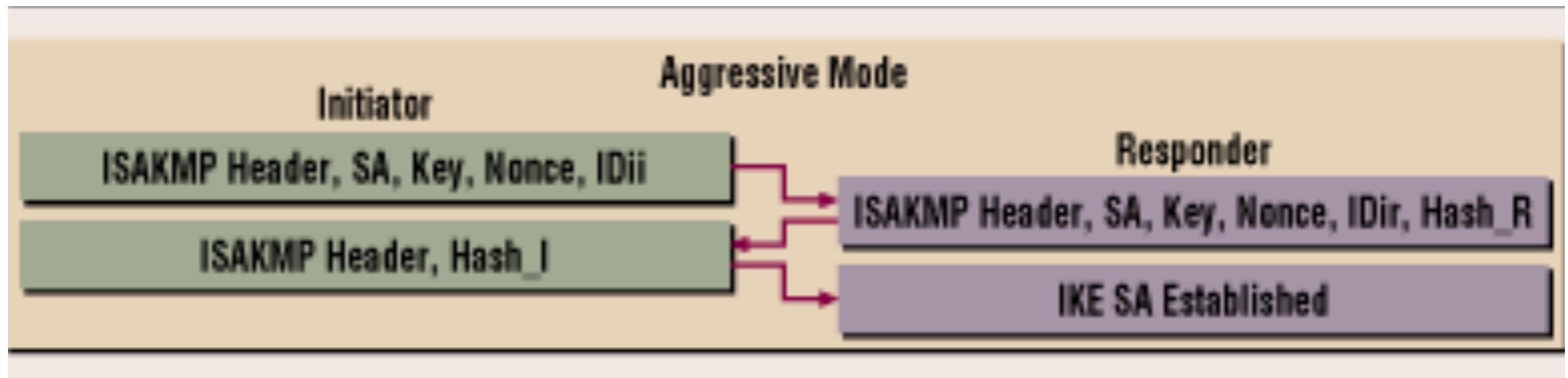
# IKE Authentication

- Pre-shared keys easy
- RSA versions I suggest you read the specifications
- X.509 good because it allows authentication based on signatures on certificate, rather than IKE agent needing full details of all peers

# IKE Aggressive Mode

- Main mode is slow, but protects the identities of the parties.
- This might not be a worthwhile trade-off
- Alternative is aggressive mode

# IKE Aggressive Mode



# Aggressive

- Packs much of the same data into three packets
- Exposes both parties' identities, as those are sent pre-key agreement (may not matter in real-world environments)
- Doesn't provide for delicate negotiation of algorithms (may not matter amongst friends).

# Phase 2: Main Mode

- Main means to establish IPsec keys
- If “Perfect Forward Secrecy” is enabled, new DH key is established each time a new IPsec key is derived (little reason to not use this)
- Three packet exchange of DH contributions, then a key confirmation packet

# Configuration Example (Racoon/KAME/ipsec-tools)

```
sainfo anonymous          Phase 2
{
    pfs_group 2;
    lifetime time 24 hour ;
    encryption_algorithm aes 192, aes 128;
    authentication_algorithm hmac_sha256, hmac_sha1 ;
    compression_algorithm deflate ;
}

remote anonymous {          Phase 1
    exchange_mode aggressive, main, base;
    my_identifier asn1dn;
    peers_identifier asn1dn;
    verify_identifier on;
    certificate_type x509 "cert.pem" "key.pem" ;
    ca_type x509 "cacert.pem" ;
    lifetime time 14400 sec;
    nonce_size 40;
    proposal {
        dh_group 5;
        encryption_algorithm aes 256;
        hash_algorithm sha256;
        authentication_method rsasig;
    }
}
```

# But needs flows defining separately

```
spdadd -n 95.46.198.9 81.187.150.211 any -P out ipsec esp/transport//require;  
spdadd -n 81.187.150.211 95.46.198.9 any -P in ipsec esp/transport//require;  
spdadd -n 95.46.198.9 81.187.150.213 any -P out ipsec esp/transport//require;  
spdadd -n 81.187.150.213 95.46.198.9 any -P in ipsec esp/transport//require;  
spdadd -n 95.46.198.9 81.187.150.210 any -P out ipsec esp/transport//require;  
spdadd -n 81.187.150.210 95.46.198.9 any -P in ipsec esp/transport//require;  
spdadd -n 95.46.198.9 81.187.150.215 any -P out ipsec esp/transport//require;  
spdadd -n 81.187.150.215 95.46.198.9 any -P in ipsec esp/transport//require;  
spdadd -n 95.46.198.9 81.187.150.217 any -P out ipsec esp/transport//require;  
spdadd -n 81.187.150.217 95.46.198.9 any -P in ipsec esp/transport//require;
```

# Or Solaris

```
# 3 days
ikesa_lifetime_secs 259200
# one hour
childsa_lifetime_secs 3600
# 51 minutes (1h - strongswan's 9m default margin)
childsa_softlife_secs 3060

{
    label "everything we do (v4)"
    auth_method cert
    remote_id ANY
    required_issuer DN="C=GB, ST=England, L=Birmingham, O=batten.eu.org private CA, OU=VPN"
    local_id DN="C=GB, ST=England, O=batten.eu.org, OU=VPN, CN=mail.batten.eu.org"
    local_addr 147.188.192.250
    remote_addr 0.0.0.0/0
    ikesa_xform { dh_group 15 auth_alg sha256 encr_alg aes }
    childsa_pfs 5
}
```

# But needs separate SA configuration (including algorithms)

```
{ laddr 147.188.192.250 raddr 81.187.150.211 } ipsec {encr_algs aes-gcm(128..256) sa shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.213 } ipsec {encr_algs aes-gcm(128..256) sa shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.210 } ipsec {encr_algs aes-gcm(128..256) sa shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.215 } ipsec {encr_algs aes-gcm(128..256) sa shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.217 } ipsec {encr_algs aes-gcm(128..256) sa shared ike_version 2}
```

# Or StrongSwan

```
conn %default
    ikelifetime=72h
    keylife=1h
    rekeymargin=9m
    keyingtries=%forever
    keyexchange=ikev2
    # group 15 for IKE
    ike=aes256-sha256-modp3072
    # group 5 for PFS
    esp=aes128-sha1-modp1536
    leftcert=cert.pem
    type=transport
    auto=start
    rightid="C=GB, ST=England, O=batten.eu.org, OU=VPN, CN=*"
    dpdaction=hold
    dpddelay=0
    mobike=no

conn mailv4
    left=95.46.198.9
    right=147.188.192.250
```

# Problems with IKE

- Very poor standardisation of implementations
  - Tunnel mode, in particular, difficult to interwork
  - Configuration files are cryptic beyond belief
  - Implementations on routers are again different

# Problems with IKE

- Relies on DH key exchanges generating “good” keys
  - Unknown quantity, and notably some classified networks use manual keying
- Distribution of pre-shared keys difficult
- Generation and signing of certificates has usual problems (using CRLs is particularly fraught)

# IKE Redux: IKEv2

- IKEv2 now becoming available, but interworking again hellish.
- Proposed 2005 (RFC 4306) but slow to get traction.
- Arrives in OSX, iOS in late 2015, used in some standalone clients before then.
- Interworking is hellish, let me restate.

# IKEv2

- Removes main/aggressive distinction, and replaces it with a four-packet exchange which authenticates IKE instances **and** agrees first keys for first SA (now CHILD\_SA).
- Standardises a lot of established but non-documented behaviour, including XAUTH and other features useful for VPNs.
- Much easier to use for a VPN directly, rather than tunnelling L2TP (or whatever) through it.
- But...

# IKEv2

- Configuration complex, and for phones often has to be done via configuration applications rather than GUIs: problematic for BYOD.
- Fewer options but still tricky to get working, and the support for VPNs gives more opportunity for incompatibility.
- Needed for mobility, amongst other things
- You can mix IKEv2 and IKEv1 on the same platform, although it can be messy

# IPsec attacks

- Not used widely outside classified networks
- Using IPsec to lift 334 network to 554 still has IL3 protection from even seeing the traffic
  - Therefore not widely attacked

# IPsec Attacks

- Basic IPsec architecture simple, robust and uses proven block ciphers
- IKE uses multiple public key mechanisms, complex key exchange, lots of difficult stuff
- My suspicion is that if you were going to attack IPsec, you attack IKE first

# IPsec Woes

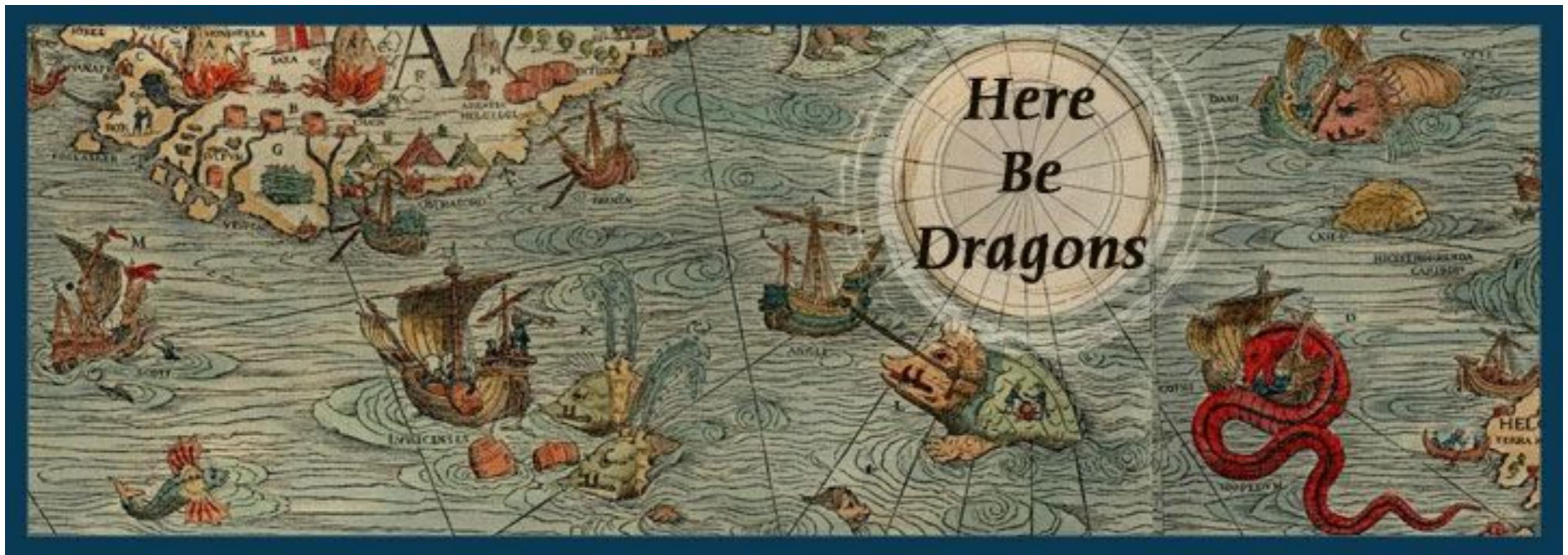
- Poor standardisation of SHA2 hash truncation, so you can agree cipher suites but it doesn't work
  - Should truncate SHA2 to half length (256->128, 512->256) but tendency to truncate everything to 160 bits , or maybe 96, or something
  - Linux <= 2.6.32 has this problem, and is still in use, so other kit perpetuates bug for compatibility
- Similar problems for GCM.
- End up using AES+SHA1 for everything
- GCM faster if you can get it working

# Network Security 19: Firewalls Redux

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# Rationale

- We skipped quickly through firewalls (ipf and iptables) without talking about syntax and operation.
- Let's talk about syntax and operation.
- And learn about the chaos of real firewall configurations in anything other than fully change-controlled secure environments



# iptables

- Let's dissect a real firewall on a real machine
- offsite6.batten.eu.org: a server which runs Cyrus replication, a few other services, and has some IPsec tunnels
- Only one set of rules: the INPUT table, called for packets headed up the stack to local processes (machine only has one interface)
- No NAT, so no mangle rules, just filter rules

# iptables

- A sequence of rules are checked for each packet as it arrives, and any matching actions are taken.
- The table is maintained by using iptables -I (upper case i) (to insert rules) or iptables -A (to append rules)

# The gory details

```
iptables -P INPUT DROP
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -N ossec
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED
    -j ACCEPT
iptables -A INPUT -i eth0 -p icmp -j ACCEPT
iptables -A INPUT -i tun0 -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 23 -j DROP
iptables -A INPUT -p esp -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 4500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --sport 17500 --dport 17500
    -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 67:68 -j DROP
iptables -A INPUT -s 128.204.195.0/24 -i eth0 -p udp -m udp --dport
    138 -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 53 -j ACCEPT
iptables -A INPUT -s 85.233.160.0/24 -i eth0 -p tcp -m tcp --dport
    25 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 123 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 443 -m state --state
    NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 80 -m state --state
    NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 53 -m state --state
    NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 3306 -m policy --dir
    in --pol ipsec -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 147.188.192.0/24 -i eth0 -p tcp -m state
    --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 104.161.79.123/32 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 149.255.32.0/24 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 46.28.203.0/24 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 5.101.172.0/24 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 69.65.45.0/24 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.17.28.0/24 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 147.188.0.0/16 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.187.150.208/28 -p tcp -m tcp --dport 22 -m
    state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 104.161.79.123/32 -p udp -m udp --dport 161
    -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -j
    REJECT --reject-with icmp-port-unreachable
.
```

```
iptables -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN  
-m hashlimit --hashlimit-upto 1/min --hashlimit-burst 1  
--hashlimit-mode srcip,dstport --hashlimit-name loglimit  
--hashlimit-srcmask 24 -j LOG  
iptables -A INPUT -p tcp -m tcp --dport 22 --tcp-flags  
FIN,SYN,RST,ACK SYN -j REJECT --reject-with  
icmp-port-unreachable  
iptables -A INPUT -m hashlimit --hashlimit-upto 1/min  
--hashlimit-burst 1 --hashlimit-mode srcip,dstport  
--hashlimit-name loglimit --hashlimit-srcmask 24 -j LOG  
.
```

# Policy

```
iptables -P INPUT DROP  
iptables -P FORWARD ACCEPT  
iptables -P OUTPUT ACCEPT
```

This is the policy for what happens if you arrive at the end of the table and haven't made a decision. Default drop INPUT, default pass FORWARD and OUTPUT (machine is configured to not forward)

# Extra Tables

```
iptables -N ossec
```

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

This is a chain that will be used to file packets against a dynamic list of bad sites maintained by the OSSEC HIDS.

# OSSEC

-S: source

-j: jump

```
iptables -A ossec -s 182.100.67.0/24 -j DROP
iptables -A ossec -s 182.118.53.0/24 -j DROP
iptables -A ossec -s 222.161.4.0/24 -j DROP
iptables -A ossec -s 58.218.213.0/24 -j DROP
iptables -A ossec -s 61.240.144.0/24 -j DROP
```

This is a list of the networks that have been making continuous SSH login attempts to my machines. After a while, I distribute the firewall list around my whole network and block the whole /24 for a day

# End of a user-created chain?

- If you haven't executed a jump to a target like DROP or ACCEPT, the flow of control returns to where the the chain was invoked
-

# Initial Acceptance

-m: module

```
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

—ctstate: connection state

The “conntrack” module has to be used explicitly with Linux firewalls (with ipf, anything whose state was kept with “keep state” is automatically accepted).

# Initial Acceptance

```
iptables -A INPUT -i eth0 -p esp -j ACCEPT  
iptables -A INPUT -i eth0 -p icmp -j ACCEPT  
iptables -A INPUT -i tun0 -j ACCEPT  
iptables -A INPUT -i lo -j ACCEPT
```

-i: interface

-p: protocol

- ESP is encrypted IPsec traffic, and cannot be processed by the firewall until it has been decoded
- tun0 is an old VPN interface where I trust the destination (no longer in use)
- Accepting everything on loopback protects against stupid mistakes

# Protocols

```
iptables -A INPUT -i eth0 -p udp -m udp --dport 500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 4500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --sport 17500 --dport 17500 -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 67:68 -j DROP
iptables -A INPUT -s 128.204.195.0/24 -i eth0 -p udp -m udp --dport 138 -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 53 -j ACCEPT
```

—dport: destination port  
—sport: source port

- 500: IKE
- 4500: IKE with NAT
- 17500: Shamefully, I can't remember
- 67 and 68: dhcp/bootp
- 138: NetBIOS (background noise)
- 53: DNS

# Protocols

```
iptables -A INPUT -s 147.188.0.0/16 -p tcp -m state --state NEW -m tcp --dport 3000 -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -p tcp -m tcp --dport 3000 -m state --state NEW -j ACCEPT
iptables -A INPUT -s 85.233.160.0/24 -i eth0 -p tcp -m tcp --dport 25 -m state \
    -state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -p udp -m state --state NEW -m udp --dport 9996 -j ACCEPT
```

- -m invokes a module, and all the following arguments are passed to that module up until the next -m. So you can use multiple modules on the same packet. Here we see a packet being checked for being the start of a new flow, and being to port 3000, where I was running a non-standard web server for NTOP.
- Netflow data for NTOP travels over port 9996 UDP.
- Permit email from one particular network only (the “ESTABLISHED” does nothing: can you think why?)

# More Protocols

```
iptables -A INPUT -i eth0 -p udp -m udp --dport 123 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -i eth0 -p tcp -m tcp --dport 4242 \
-m state --state NEW,ESTABLISHED -j ACCEPT
```

Again, ESTABLISHED does nothing

NTP, http/s, ssh, DNS over TCP, marking undergraduate exercises

# Slow logging of attacks

```
iptables -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN \
-m hashlimit --hashlimit-upto 1/min --hashlimit-burst 1 \
--hashlimit-mode srcip,dstport --hashlimit-name loglimit \
--hashlimit-srcmask 24 -j LOG
```

```
iptables -A INPUT -m hashlimit --hashlimit-upto 1/min --hashlimit-burst 1 \
--hashlimit-mode srcip,dstport --hashlimit-name loglimit \
--hashlimit-srcmask 24 -j LOG
```

–tcp-flags looks at the first set of flags, and matches if any of the second set of flags are set; here we are looking for first packet on connection  
hashlimit module maintains a hashtable of addresses and ports, and rate-limits based on new entries

# IPsec Policy

```
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 3306 \
-m policy --dir in --pol IPsec \
-m state --state NEW,ESTABLISHED -j ACCEPT
```

Accept MySQL, but only over IPsec

# “Friendly Machines”

```
iptables -A INPUT -s 147.188.192.0/24 -i eth0 -p tcp -m state  
    \-state NEW,ESTABLISHED -j ACCEPT  
iptables -A INPUT -s 147.188.192.0/24 -p tcp -m state  
    \-state NEW,ESTABLISHED -m tcp --dport 161 -j ACCEPT  
iptables -A INPUT -s 147.188.192.0/24 -i eth0 -p udp -m udp --dport 161 -j ACCEPT
```

All TCP from bham-cs server network  
All SNMP (tcp and udp) from same

# ipf

- Here's a more complex scenario
- mail.batten.eu.org, my primary mail server
- Used to be a global zone with shared IP, hence all the /30 netmasks
- Now a local zone

# The file (1)

```
block in quick from pool/100 to any

# pass outbound traffic and replies to it
pass out quick on mailprod0 proto tcp from 147.188.192.248/30 to any flags S/SA keep state
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any port = 123
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any keep state
pass out quick on mailprod0 proto icmp from 147.188.192.248/30 to any keep state

pass in quick on mailprod0 proto ipip from 81.2.79.220 to any
pass out quick on mailprod0 proto ipip from any to 81.2.79.220

pass in quick on ip.tun0
pass out quick on ip.tun0
pass in quick on log1
pass out quick on log1

# shut down stray TCP connections and block all other traffic

block return-rst out quick proto tcp all
block out quick all

# zap rfc1918 and so on: no quick on pass because these are first-pass rules

pass in on mailprod0 all head 100
block in quick from 192.168.0.0/16 to any group 100
block in quick from 172.16.0.0/12 to any group 100
block in quick from 10.0.0.0/8 to any group 100
block in quick from 127.0.0.0/8 to any group 100
block in quick from 169.254.0.0/16 to any group 100
block in quick from 224.0.0.0/3 to any group 100
```

# The file (2)

```
# block broadcast and multicast

block in quick from any to 147.188.192.255/32
block in quick from any to 224.0.0.0/3

# accept SMTP connections from known MXes

block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    port = 25 flags S/SA head 101
pass in quick from 147.188.128.54/32 to any keep state group 101
pass in quick from 147.188.128.127/32 to any keep state group 101
pass in quick from 147.188.128.129/32 to any keep state group 101
pass in quick from 147.188.128.219/32 to any keep state group 101
pass in quick from 147.188.128.221/32 to any keep state group 101
pass in quick from 147.188.192.250/32 to any keep state group 101
pass in quick from 147.188.192.249/32 to any keep state group 101

# generally acceptable protocols; block quick because there is no further TCP
# processing

# imaps, submission, domain, imap, http, https, ssh

block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    flags S/SA head 102
# 5877 is obsolete clone of 587,
# 993 is obsolete imaps prior to use of STARTTLS verb
# block return-rst in quick from any to any port = 993 keep state group 102
block return-rst in quick from any to any port = 5877 keep state group 102
pass in quick from any to any port = 587 keep state group 102
pass in quick from any to any port = 993 keep state group 102
pass in quick from any to any port = 53 keep state group 102
pass in quick from any to any port = 143 keep state group 102
pass in quick from any to any port = 80 keep state group 102
pass in quick from any to any port = 443 keep state group 102
pass in quick from any to any port = 22 keep state group 102
pass in quick from 128.204.195.144 to any port = 2010 keep state group 102
pass in quick from 128.204.195.144 to any port = 2007 keep state group 102
pass in quick from 128.204.195.144 to any port = 2009 keep state group 102
pass in quick from 128.204.195.144 to any port = 2003 keep state group 102
```

# The file (3)

```
# and some UDP we need; again block quick as there is no further UDP processing
# ntp and domain

pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 123
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 53 keep state
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 1514 keep state
block return-icmp in quick on mailprod0 proto udp from any to 147.188.192.248/30
block in quick on mailprod0 proto udp all

# management machine can ping us

pass in quick on mailprod0 proto icmp \
    from 147.188.130.98/32 to 147.188.192.248/30 \
    icmp-type echo keep state

# backstop: send a reset for non-S/SA TCP traffic, discard all of it, with no logging
# (stray S/SA frames have been caught and logged higher up)
# logging for everything else

block return-rst in quick proto tcp all
block in log level local1.info quick all
```

# “quick”

- ipf logic is to keep on making matches, recording the action that is required
- When it reaches the end of the rules, it takes the action from the last match
- quick means “done, take this action now”

# OSSEC, again

block in quick from pool/100 to any

ipf “pools” are lists of addresses that are  
matched together in one rule

Used in this case to record the long-term bad  
sites and block them

# Outbound Rules

```
# pass outbound traffic and replies to it
pass out quick on mailprod0 proto tcp from 147.188.192.248/30 to any flags S/SA keep state
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any port = 123
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any keep state
pass out quick on mailprod0 proto icmp from 147.188.192.248/30 to any keep state

pass in quick on mailprod0 proto ipip from 81.2.79.220 to any
pass out quick on mailprod0 proto ipip from any to 81.2.79.220
```

S/SA = Syn is Set, Ack is not set

keep state = accept anything that looks like a reply

ipip is a tunnel

# Internal Networks

```
pass in quick on ip.tun0
pass out quick on ip.tun0
pass in quick on log1
pass out quick on log1
```

ip.tun0: result of de-encapsulating ipip  
log1: private network into another zone

```
log1: flags=100001000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,PHYSRUNNING> mtu 9000 index 4
      inet 192.168.106.101 netmask ffffff00 broadcast 192.168.106.255
```

# Default Block Out

```
# shut down stray TCP connections and block all other  
# traffic  
  
block return-rst out quick proto tcp all  
block out quick all
```

Any packets exiting the machine “out of state” generate an RST back to the calling process (can you think why?)

Drop everything else

# Block illegal packets

```
# zap rfc1918 and so on: no quick on pass because these are first-pass rules  
pass in on mailprod0 all head 100  
block in quick from 192.168.0.0/16 to any group 100  
block in quick from 172.16.0.0/12 to any group 100  
block in quick from 10.0.0.0/8 to any group 100  
block in quick from 127.0.0.0/8 to any group 100  
block in quick from 169.254.0.0/16 to any group 100  
block in quick from 224.0.0.0/3 to any group 100
```

Drop packets on mailprod0 which appear to come from places that they shouldn't be coming from

# Multicast and Broadcast

```
# block broadcast and multicast  
block in quick from any to 147.188.192.255/32  
block in quick from any to 224.0.0.0/3
```

I'm on a shared network, but don't need to listen to any particular protocols to operate

Multicast isn't being used and can be dangerous

# Mail from known relays

```
# accept SMTP connections from known MXes

block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    port = 25 flags S/SA head 101
pass in quick from 147.188.128.54/32 to any keep state group 101
pass in quick from 147.188.128.127/32 to any keep state group 101
pass in quick from 147.188.128.129/32 to any keep state group 101
pass in quick from 147.188.128.219/32 to any keep state group 101
pass in quick from 147.188.128.221/32 to any keep state group 101
pass in quick from 147.188.192.250/32 to any keep state group 101
pass in quick from 147.188.192.249/32 to any keep state group 101
```

# Other TCP

```
block return-rst in log first level local1.info quick \
  on mailprod0 proto tcp from any to 147.188.192.248/30 \
  flags S/SA head 102
# 5877 is obsolete clone of 587,
# 993 is obsolete imaps prior to use of STARTTLS verb
# block return-rst in quick from any to any port = 993 keep state group 102
block return-rst in quick from any to any port = 5877 keep state group 102
pass in quick from any to any port = 587 keep state group 102
pass in quick from any to any port = 993 keep state group 102
pass in quick from any to any port = 53 keep state group 102
pass in quick from any to any port = 143 keep state group 102
pass in quick from any to any port = 80 keep state group 102
pass in quick from any to any port = 443 keep state group 102
pass in quick from any to any port = 22 keep state group 102
pass in quick from 128.204.195.144 to any port = 2010 keep state group 102
pass in quick from 128.204.195.144 to any port = 2007 keep state group 102
pass in quick from 128.204.195.144 to any port = 2009 keep state group 102
pass in quick from 128.204.195.144 to any port = 2003 keep state group 102
```

Look at SYN packets, match against list of valid ports (and in some cases sources)

# Other UDP

```
# and some UDP we need; again block quick as there is no further UDP processing
# ntp and domain

pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 123
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 53 keep state
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 1514 keep state
block return-icmp in quick on mailprod0 proto udp from any to 147.188.192.248/30
block in quick on mailprod0 proto udp all
```

In 2015 NTP firewall should be tighter, but in this case it is legacy: mail.batten.eu.org is now a zone and Solaris zones don't have their own clock, so don't run NTP.

“return-icmp” sends an ICMP port unreachable.

# University management machine (may be legacy)

```
# management machine can ping us  
pass in quick on mailprod0 proto icmp \  
from 147.188.130.98/32 to 147.188.192.248/30 \  
icmp-type echo keep state
```

Passes ICMP echo requests (the active side of a PING) from one particular machine, and gets ready to send a response

# Default Deny In

```
block return-rst in quick proto tcp all  
block in log level local1.info quick all
```

All that is logged is obscure  
protocols that have been dropped,  
mostly for debugging purposes

# Summary

- Firewalls suffer from bitrot: they accumulate “stuff” which you don’t understand but don’t dare remove
- Their syntax is usually complex, and the effects subtle and hard to debug
- Tendency to fiddle with a working one, rather than start from scratch

# Network Security 19: Practical VPNs

[i.g.batten@batten.eu.org](mailto:i.g.batten@batten.eu.org)

# Practical VPNs

- We've talked about VPNs in principle
- We've talked about IPsec in detail
- What are the real protocols?

# Practical VPNs

- SSL VPN
  - OpenVPN
- L2TP + IPsec
- “Cisco style” IPsec VPN

# SSL VPN

- Covers several alternatives
  - “Client-less” systems provide access to http resources via an SSL connection between the browser and the VPN server
  - Richer systems use the same connection and a light-weight, often Java, client which does tunnelling

# Client Less

- You can argue this is not really a VPN
  - User connects to <https://sslvpn.my.com>, authenticating once
  - Server presents various corporate resources, all behind sslvpn.my.com (and its certificate)
  - Connection is re-used, and resources are fetched over it, even if they aren't really https-enabled
    - Effectively, the SSL VPN server is a reverse proxy

# Client Less Problems

- Causes all corporate resources to appear to come from a single domain as far as browser security is concerned
  - sslvpn.my.com/mail
  - sslvpn.my.com/expenses
- Massive potential for cross-site scripting, fun and games with cookies, etc, if any one of the resources is compromised.

# This is fixable

- With care, and a lot of assurance, you could divide corporate resources into sub-domains, so that browser security sees them as distinct
- If you are smart and determined enough to do this, you are smart and determined enough to do something better

# SSL Forwarding

- Client downloads a Java app (or similar), which runs in the browser
- Granted elevated privileges in order to listen on 127.0.0.1:port
- Applications talk to 127.0.0.1:port and are tunnelled to server:port.
- Like ssh port forwarding for people who can't cope with ssh port forwarding

# OpenVPN

- Uses authentication mechanisms from SSL, so leverages OpenSSL (used as a library) functionality
- Runs over TCP or UDP so is OK for firewalls, and has proxying extensions
- But fully-featured packet-based VPN which looks like an interface to the computers involved

# OpenVPN

- Client requires openvpn client (userspace) and “tun” or “tap” drivers (kernel-side, now shipped with almost every Unix-alike including OSX, Linux and Solaris).
- Server requires suite of daemons, plus similar kernel support.
- Modern versions can do 6-in-6, 6-in-4 and 4-in-6 as well as 4-in-4.
- Implementations available for assorted routers, and commercial virtual appliances.

# OpenVPN v SSL VPN

- OpenVPN requires client software in all situations, and installing that software can be an adventure (the iOS stuff is particularly annoying, as it sits outside the OS VPN framework)
- SSL VPN can run clientless, although that has problems
- SSL VPN clients self-install from target address, and in some cases do not need admin password.

# OpenVPN Benefit

- OpenVPN can be configured to retain a copy of the **exact** certificate presented by the server
- Means attacker who can forge signed certificates with arbitrary subjects cannot perform MitM
- SSL VPNs usually just use certificates in the normal way, so attacker who can get a certificate in the right name can pose as server

# L2TP

- L2TP = Layer 2 Tunnelling Protocol
  - Successor to L2FP (Cisco) and PPTP (Microsoft)
  - Sends packets as UDP (so passes through NAT) containing tunnelled data
  - Commonly used to tunnel PPP

# PPP

- Point to Point protocol
- Derived from HDLC
- Mechanism for sending IP packets down serial lines and other point to point (note: not peer to peer) links
  - Replaced SLIP (Serial Line IP) as the protocol of choice for modems
- Also used for tunnelling elsewhere, cf. PPPoE, PPPoA.

# Why PPP?

- PPP has its own authentication mechanism, permitting username/password login as part of setting up a connection
- Also has mechanisms for negotiating MTU, IP Addresses, etc
  - Hence use by ISPs
  - Incoming connections usually handed to a Radius server which authenticates and issues IP numbers

# PPP over L2TP

- Raw PPP frames can be encapsulated into L2TP frames
- They arrive at the other end and are removed from the encapsulation
- Overall effect is as though there were a piece of wire between the two points, carrying PPP

# Security?

- PPP has no encryption
- L2TP has no encryption (PPTP did, but it was rubbish).
- That looks like a problem, doesn't it?
- IPsec to the rescue

# L2TP/IPsec

- L2TP appears to the network as a flow of UDP packets between the client and the server
- This is ripe for securing with IPsec
- Usual method is using pre-shared keys, but certificates can be used

# L2TP/IPsec

- Presented to users as a “group secret” and then their own username and password
  - **NB: this is the only thing protecting all users against a MITM which will yield their personal credentials.**
  - Reality (and getting it working on Mikrotik routers!) is rather more complex
  - Group Secret is a pre-shared key for IKE (so certificates can be used instead, for the very keen)

# L2TP/IPsec

- Client talks IKE to server to establish session keys, secured with group secret common to all users of the VPN server
  - Some VPN client software goes to great lengths to hide this secret from users
- Server and client establish an SPI for L2TP packets
- Client makes PPP connection over L2TP to server, protected by IPsec

# L2TP/IPsec

- Group key provides (some) confidence client is talking to the right server, and that the connecting client is not just some random machine on the Internet
- Privacy comes from Diffie-Hellman negotiation of a session key (forward secrecy in event of later compromise of secret)
- PPP login/password proves user is valid and allows per-user profiles (and can use OTP)
- PPP login is protected by IPsec confidentiality

# L2TP/IPsec

- Widely available, standard on Windows, Android, OSX and iOS. Components usually present for other systems (although can be complex to set up, as involve merging PPP and IPsec)
- Lots of Appliances available
- Tends to use IKE aggressive mode, hence needs pre-establishment of crypto suite in use (normally 3DES).
  - Can be difficult in heterogeneous environments

# “Cisco” IPsec

- Instead of complex stack of PPP over L2TP over IPsec, why not just use IPsec tunnel mode?
- Answer: lack of standardised authentication

# IPsec XAuth

- Standard IPsec authentication mechanisms include pre-shared secrets and certificates of various sorts.
- Very much aimed at host-to-host security, rather than user-to-host
  - Doesn't support any sort of two-factor or challenge/response authentication

# IPsec XAuth

- Closest analogue to a password, the pre-shared key, is used directly as a key
  - Needs to be long and random, although you can imagine use of a password derivation function, but also...
  - Both sides needs copies in plaintext, which may not be acceptable
  - No easy way to use to support OTP

# IPsec XAuth

- Provides mechanism for a sequence of messages passing arbitrary requests (including nonces) and getting arbitrary responses (including hashes)
- All can then be passed to/from a Radius server
- Annoyingly, weakly standardised and full of Proprietary extensions
  - Supported directly in IKEv2, but transition is slow

# “Cisco” IPsec

- Historically “Cisco VPN Client”
- Now “Cisco Anyconnect Secure Mobility Client”
  - Bundled with iOS and OSX, presumably others
  - Cisco logo only third-party branding on iPhone; rumoured to be part of deal over iOS v IOS.
- XAUTH supported by IOS (Cisco operating system) and therefore on various Cisco appliances as well as full-feature routers
- Reverse engineered onto racoon and charon (helped by old RFCs) but only for the enthusiastic
- IKEv2 versions are standardised

# VPN Deployment

- VPN tends to work on the assumption that the VPN secures the connection, no-one else needs to worry about it
  - SSL VPNs open up a range of cross-site attacks if resources from different trust domains are aggregated
  - All VPNs provide trusted paths deep into the enterprise, to applications that are not secured
  - Two factor, two factor, two factor

# VPN Summary

- SSL VPNs work for their problem space, but get messy and complex for arbitrary applications.
- OpenVPN works, but lack of commercial support on client side an issue
- L2TP/IPsec works well and is supported, but complex stack and difficult to diagnose problems in complex networks
- Cisco stuff easy if your clients are supported, almost impossible if they aren't; requires effectively proprietary extensions to IKEv1.
  - But Cisco support, eg, Solaris on SPARC!

# Network Security 21: Putting it Together, Taking it Apart

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# We've looked at...

- Host security (firewalls and attack-surface minimisation)
- Firewalls
- IDS
- VPN
- IPsec

# We've looked at...

- Proxies
- Content Filtering
- Wireless Security
- Two Factor Authentication
- DNS Attacks

# We haven't looked at

- Virus scanners (run mostly on hosts, but can run on web proxies and mail gateways)
- DNSSec (lack of wide deployment and unclear what threats it mitigates)
- Protocol Design in detail (different course!)

# Pulling from other courses/ knowledge

- NAT provides some security
- Secure applications are safe to expose to the internet
- We obviously need asset registers, risk registers and so on in order to design bespoke security solutions (TM)

# Put yourself out of a job!

- 2005-style, perhaps even 2015-style, every company with a substantial web presence ran mail, web, RAS/VPN, perhaps e-commerce, perhaps e-payment.
  - Either developed in house, often by unskilled staff, or bought in and modified, or occasionally bought as a managed service.

# Put yourself out of a job!

- 2018-style, forward-looking IT functions buy as much as possible as services, not products.
  - Who's running their own mail?
  - Who's running their own web service?
  - If your applications are in the cloud, like Salesforce and Google Apps / Office 365, do you need a VPN?

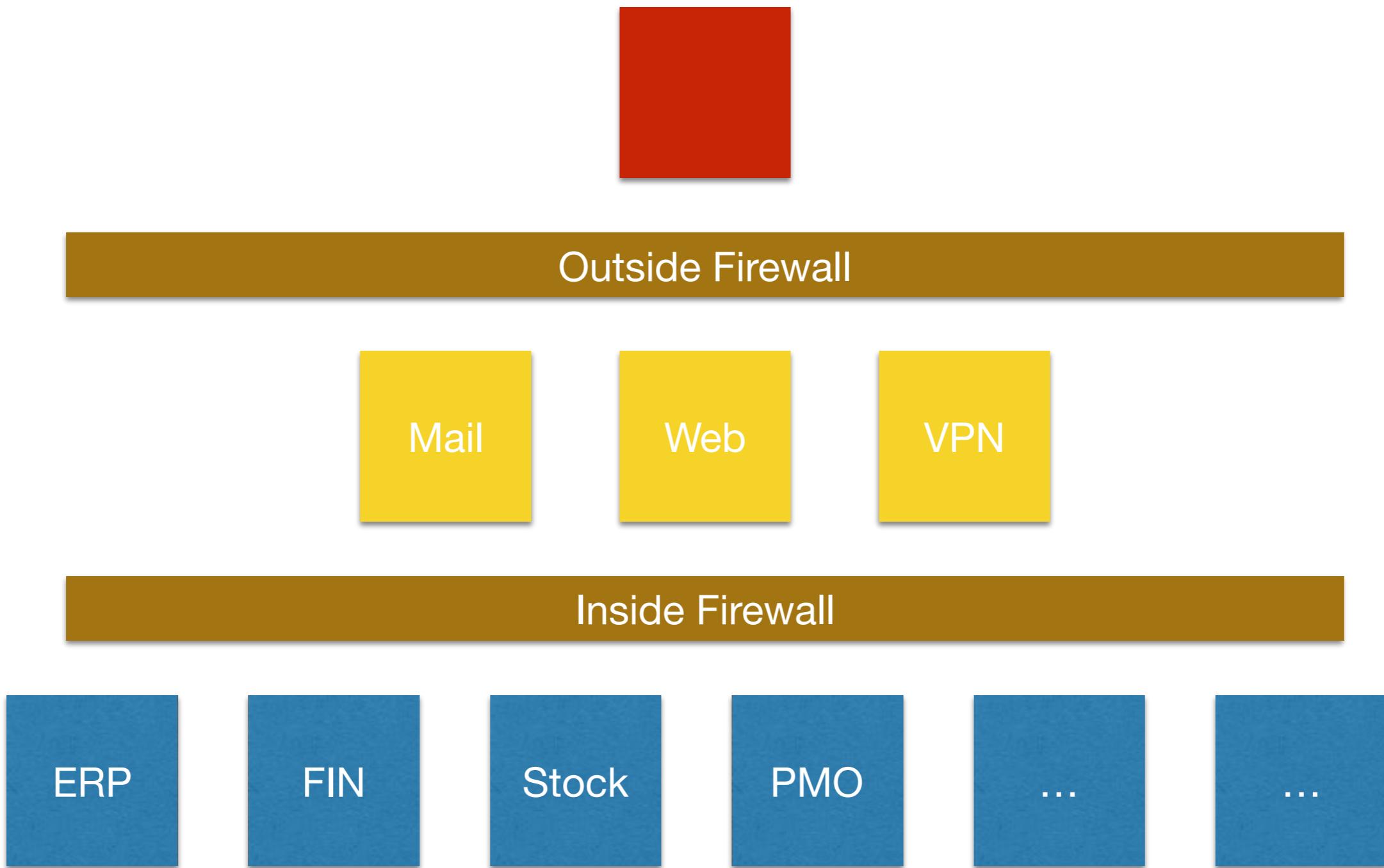
# Data Centres

- No longer certain that companies will have data centres
  - 10TB of storage fits in a shoebox
  - And you can rent it from Amazon for a few hundred dollars a month anyway

# Web Applications

- Common model is that web applications are developed to one standard for use externally, a lower standard for in-house only use, protected by VPN (SSL-VPN is enough).
  - Why is this acceptable? Why not do everything to an “Internet” standard?

# Old Design Pattern



# Bring Your Own Device (BYOD)

- Two choices for a business
  - Staff bring their own kit and you manage the risks in an appropriate manner in collaboration with your auditors and other stakeholders
  - Staff bring their own kit anyway, and use it anyway, and you don't get a say
  - Senior and middle management **will not back you** if you attempt to ban iPads.

# Home Working

- Ability to isolate home devices reducing
  - people won't accept devices that don't split tunnel
- People want full functionality from home
  - including telephony, printing, CTI, etc
  - IT Directors and CSOs that say “no” will be saying “no” to their recruitment agency.

# Financial Reality

- When laptops were expensive, only senior management and salesmen had laptops
  - As IT Director in all but name, I didn't get a laptop until about 1997, and (from memory) at meetings in Silicon Valley in 1995–2000 laptops were nothing like universal for technical people (to be fair, these were technical meetings at server vendors!)
  - Now, everyone either has a laptop from work or owns a laptop/iPad at home, and laptops are widely issued: massive capital cost

# “Generation Y”, Digital Natives, etc

- For people under 30, their digital life **is** their life.
- Boundaries between work and play blurred
- Portfolio of digital skills one of the things they are selling
- Companies that issue large black Windows 10 (or 7!) machines in locked-down state
  - a.not making best use of staff skills
  - b.not best at recruiting staff
- If you want the best people, you need the best environment

# Take-away quote

- **In five years' time, we'll only be issuing laptops to people to whom we issue trousers**
- Essential to issue locked-down machines to emergency and uniformed services, plus receptionists, field engineers, some call centres, etc.
- And people will also use supplied systems (dealing floor displays, CAD, HPC, etc).
- But for many staff, they will also have a personal machine on their desk or in their hand, and that machine will blend work and personal.

# As go computers, so go phones. Double.

Pat Metheny & Lyle Mays



As Falls Wichita,  
So Falls Wichita Falls

- Many companies still issuing Blackberry devices (in particular)
- No appetite amongst staff for this, everyone wants to have their work phone be their personal phone (or their personal phone be their work phone)
- Phones now join internal WiFi networks

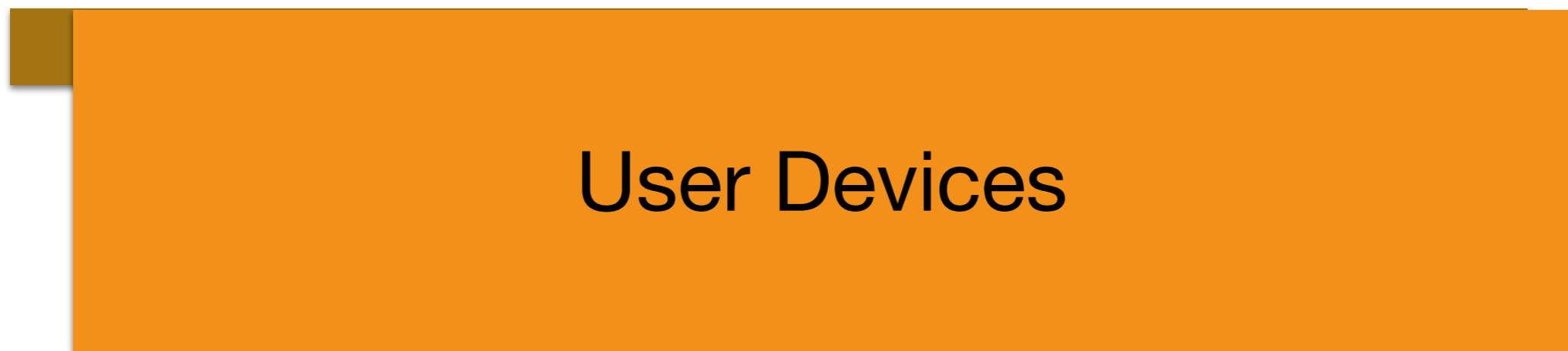
# BYOD has to happen

- In universities, *de facto* already here: few laptops purchased and centrally managed for academics, every academic has a laptop
- Ditto development shops, start-ups, tech companies
- Corporates still exercising control but cannot last forever

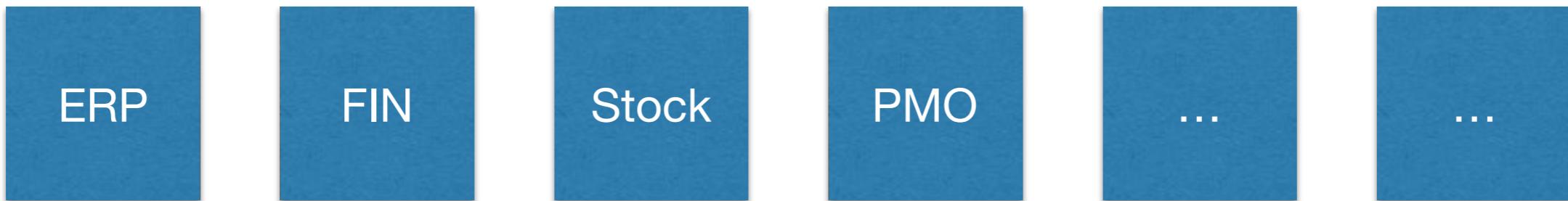
# Consequences for Network Security

- Take five minutes, in groups of two or three, and come up with three issues which arise from people having a laptop/phone device on the corporate network which they control.

# Borderless Networks



Inside Firewall



ERP

FIN

Stock

PMO

...

...

# What should we do?

- Take another five minutes and come up with three improvements we need to make to standard network architectures.

# Implications

- Host security
- Single Sign On / Two Factor
- Virtualisation
- “Islands”
- Networking

# Host Security

- All machines are exposed to the Internet, in the form of user devices which have Internet connectivity
  - In reality, has been true for ten and more years
  - On campus, and in large offices, not inter-penetrated with third-party WiFi.
  - Hosts containing sensitive data have to be secured, as in the first few lectures.

# Single Sign On / Two Factor

- Devices are going to store corporate credentials, and those credentials will be useful remotely
  - Used to joke we could safely publish root password on front page of newspaper: not any more.
- Two factor with hardware devices will become critical
  - But is cumbersome
  - SSO was the big corporate “thing” ten years ago, then died a messy death. Has to be solved.

# Virtualisation

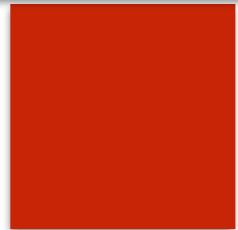
- You're running Linux and Solaris instances to get access to particular facilities on a range of devices
- One option for companies is to ship a single build of the corporate environment as a virtual machine, for users to run on their own systems
  - Different security model, for example attacks in the network layer of the host system
  - Not perfect, but a good transition aid

# “Islands”

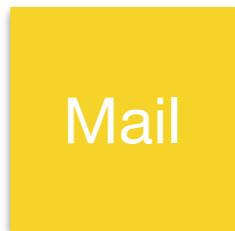
- In large enterprises, the “outer, proxy, inner, secure” model may be replicated in miniature.
- Secure offices, secure cells within the organisation
- Access by VPN servers dedicated to the island

# Islands

User Devices



Outside Firewall



Outside Firewall



Inside Firewall



# Networking

- Much more need for VPNs (remote working, distributed workplaces)
- Architecture of those VPNs much harder (endpoints not trusted)
- IPsec may also be needed as data centre moves to cloud

# Issues to Ponder

- Territoriality (legal actions, interception, search warrants, Anton Pillar orders)
- Data Protection (US rules considerably looser than ours, German rules considerably tighter)
- Accountability (if you have a contract with Amazon and it goes wrong, is the money enough?)
- Auditability (Sarbox, JSox, SSAP, ISO 27001, etc)

# Network Security 22: The Law

[i.g.batten@bham.ac.uk](mailto:i.g.batten@bham.ac.uk)

# UK/EU Centric

- European Convention on Human Rights
- European Data Protection Directive
  - Transposed into most EU members
- European Data Retention Direction
  - Ditto
- Most countries have laws like our Regulation of Investigatory Powers 2000 legislation and the (as yet untested) Investigatory Powers 2016 which came into force at the end of 2016.

# Basic Rights

Article 8 – Right to respect for private and family life

- 1. Everyone has the right to respect for his private and family life, his home and his correspondence.**
- 2. There shall be no interference by a public authority with the exercise of this right except such as is in accordance with the law and is necessary in a democratic society in the interests of national security, public safety or the economic well-being of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and freedoms of others.**

# UK Law

- ECHR now adopted into UK law by Human Rights Act 1998
- Even if we abolished that, we would still be signatories to the ECHR.
- Not related to the EU: leaving the ECHR much more serious, as would involve leaving Council of Europe.
  - But in the current febrile climate, who knows?

# IOCA 1985

- Historically, interception was done on a “nod and a wink” basis. All telecoms controlled by government (GPO) so police and agencies just spoke to their fellow civil servants. Regulation honoured more in the breach.
- Interception of Communications Act 1985 introduced when GPO privatised and OLOs started to appear (Mercury, initially)
  - Put interception on a statutory footing.

# Be down with telecoms kids

- LO: licensed operators (usually taken to mean big incumbents)
- OLOs: other licensed operators
- MOLOs: mobile licensed operators.
- In the US, ILEC (incumbent local exchange carrier) and CLEC (competitive local exchange carrier) roughly equivalent to LO and OLO.
- In UK, competitive local carriers may or may not offer voice: SMPF (Shared Metallic Path Facility) means you can be connected to Talk Talk's DSLAM but BT's voice facility.

# Case Law

- Halford v. United Kingdom , (20605/92) [1997] ECHR 32 (25 June 1997)
  - Police used powers of interception to listen to communications between employee and her lawyer, prior to industrial tribunal case on sex discrimination
  - ECtHR gave right of privacy to certain calls made on business premises

# Data Protection Act 1998

- Imposed obligations on fair processing (not part of this course)
- But Seventh Data Protection Principle applies:
  - Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.

# RIPA 2000

- Regulation of Investigatory Powers Act 2000
- Response to Halford case, and other events
- I'm biased for reasons I will explain, but I think it is basically sound legislation
- Accusations of being “snoopers' charter” forget that prior to RIPA all this stuff was just done anyway
- Never about terrorism: about entire intercept regime, to balance Article 8 rights with IOCA capabilities.

# RIPA CoP

- Provides extensive guidance on what you can do on your own networks if you are not a telecoms operator.
- Limited case law.
- Halford issues over private phonecalls (etc) rendered moot by mobiles
- Subject to notification, you can pretty much look at anything you want

# RIPA in Enterprise

- Public sector organisations tend to err on side of caution, and are wise to have internal approval system for access to employee/student data
- Private sector tends to say “our wires, our bits” (and puts that in contracts of employment)
- There will be complex case law over BYOD (see last lecture) because it’s not obvious what is “our wires” in that context: be prepared for excitement.

# RIPA/IPA in operators

- Operators should not look at content without reasonable cause (essentially, without a warrant)
  - Not clear that an offence is actually committed, as RIPA is all about government powers, not citizen protection, but common-law duties of confidence probably apply
  - If an operator looks at your packets, what happens?
- Distinction for warrants between communications data and content

# Communications Data

- Headers, envelopes, logging, billing...everything that isn't the actual content of the communication
- Available to police, agencies and a cast of thousands (the Egg Marketing Board!) over the signature of a senior officer.
  - Single Point of Contact regime should be followed
- Mostly (until recently almost exclusively) reverse DQ

# Content Data

- Requires warrant personally signed by minister
  - Home Secretary for most cases, Foreign Secretary for GCHQ work not affecting UK citizens
- Worth noting: nothing in Snowden revelations suggests policy is being broken, but much evidence that they are sailing very close to the wind
  - Question: is it OK to intercept a whole feed, filter with a computer and get a warrant for the output? Currently, probably yes, but this is at the heart of the debate about bulk collection. Case law is needed.

# Responsibility

- Contrary to popular belief, operators are not responsible for data on their networks, nor are enterprises, with some very narrow exceptions
  - Sexual Offences Act provides some protection for enterprises
  - Running open WiFi is perfectly OK
  - Salesmen love to say otherwise
  - Very narrow list of “illegal to possess” material is a concern for enterprises, but DPA/Police are taking a pragmatic approach.

# Retention of Comms Data

- Since 2001, operators can “voluntarily” retain communications data for longer than they themselves need it (codified by EU directive 2006/24/EC, as waiver on Data Protection principles; S.28 and S.29 of DPA 1998 also provide general protection).
- Data Retention and Investigatory Powers Act 2014 puts this on a statutory footing, with S.1 notices allowing SoS to mandate retention
  - Probably mainly DHCP/Radius logs, but we don’t know
  - In principle every packet header
- NB: does NOT permit retention of content, or permit SoS to mandate retention of content.
- More retention (definitely packet headers, cf. Internet Connection Records) coming in IP 2016, but we haven’t seen the codes of practice in detail yet.

# Jigsaw Attacks not Covered

- Current Information Commissioner guidance does not regard as sensitive two independent data sets which are each themselves not sensitive, but when joined are
  - eg, health records with name replaced by nonce, and a second set of records mapping name to nonce.
- Also not obvious that identifiable anonymised data (postcode + DoB is unique, for example) is covered.
- Result: headers of IP packets not under DPA, even if links mylaptop.bham.ac.uk and embarrassing.website.com.

# The care.data debacle

- Common problem is over-estimating impact level of data you hold (cf. companies claiming to have “IL5”, which they almost always don’t)
- However, bad things happen when you under-estimate impact level or public concern
- care.data: database of all hospital events and GP interactions, as a research and commissioning tool: not aggregated, but anonymised
- Parliamentary Evidence session was a complete car crash, and culminated with ministerial apology for misleading parliament
- Mixture of health-naive IT people and InfoSec-naive health people conspired to make fools of each other, and the project was canned.
  - Which is a shame, as the research would have been valuable. Sign up to UK.Biobank, please.
- Hammers home need to understand the data you are holding, its legal status, and the reputational risk of disclosure.

# This all makes firewalls OK

- Have been arguments that firewalls might contravene DPA!
- Not valid.
- However, DPI almost certainly does: look up the debacle of Phorm.

# Unsolved Problems

- Virus Scanners look at content
- Spam Filters do as well (in most cases)
- Are these interception per RIPA 2000? Are these processing per DPA 1998?
  - Confusion over this led to problems with voicemail's status and collapse of Hacking Trial investigations
- “Doormat principle”: once it’s on your doormat, it needs a search warrant, not an interception warrant.
  - Where is your electronic doormat?

# Intercept Readiness

- Distinction between operator and enterprise much more slippery than it used to be
  - WhatsApp are a telecoms operator in every obvious way, but aren't an OLO. Not regulated by Ofcom, which means fewer obligations. *A fortiori* for Skype.
- RIPA 2000 allows installation of “black boxes” and has a fairly flexible definition of what a telecoms operator is. IPA 2016 extended this.
- Ability to provide a feed of data from a core switch might save a lot of grief, unless you have a taste for prison cells.

# Intercept Readiness

- Old-school telcos maintained cell of people with DV clearance who were employees, but able to action intercept on behalf of government (list of people for whom there are active intercept warrants is classified, for obvious reasons).
- Not obvious that is the case now, and intercept may need to be done without operator knowing who is being intercepted, or with government not trusting operator to hand over take correctly.
  - In any event, it should not be possible for random operator employees to discern volume/targets.
- Almost no intercept capability in DSLAMs and other exchange-premises equipment: purely Metro/core switching.

# Intercept Readiness

- Commercial status of voice is unclear, but core voice switches are **old** technology: UK is all System X (70s, manufactured 80s) and small amounts of 5ESS and AXE10 (similar age design, more modern). Anything sold in UK or US has intercept as a feature (particularly System X, also preference working and massive overload control).
  - Has anyone spoken to Telent about jobs?
- “Soft Switches” inherit feature set from older equipment (or not, which is why they often fail: 999 overload control is hard)
- So voice intercept with appropriate security and handover well-embedded.
- There might be a market for data products with similar functionality for data: at the moment it’s done by filtering ilk intercept.

# Little Case Law

- Operators aren't interested in fighting The Man on behalf of their customers.
  - Except for A&A and other “boutique” operators.
- 1998: median customer young, liberal and sceptical. No major terrorist threat directed at west (in UK, “the troubles” pretty much over).
- 2018: median customer older, less liberal, more trusting, world rather more unstable
  - Snowden revelations no big deal outside Guardian readers. Agencies largely trusted.

# Regulation

- UK government regime overseen by Interception of Communications Commissioner, retired High Court Judge with appropriate clearance, plus small (but now more adequate) staff
  - Establishment to his fingertips, but has been fairly robust over the years
  - High Court Judges not easily pushed around, retired ones even less so

# Regulation

- Activity by companies unregulated, and only subject to oversight if Information Commissioner becomes involved (or if an employee or customer brings a civil action)
  - IC now issuing big fines
- Bad behaviour by employer might be cited at ET, but no evidence of it happening (unfortunately, ETs are “unreported” and don’t set precedent: nothing at an EAT yet)

# Conclusions

- Enterprise: OK to look at communications data, content data also OK provided you have some loose governance. AUP protects you. Theoretical ET issue, as yet untested.
- Government: definitely not OK to look at anything without following process
- Operator: content data definitely not OK, use of communications data must be fair and proportionate (DPA).
  - You aren't registered to process your customers' data, so cannot without a letter of comfort from government.

# Outside the UK?

- I'd love to hear other countries' experience.