Security 15: DoS, Amplification, DNS Attacks

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

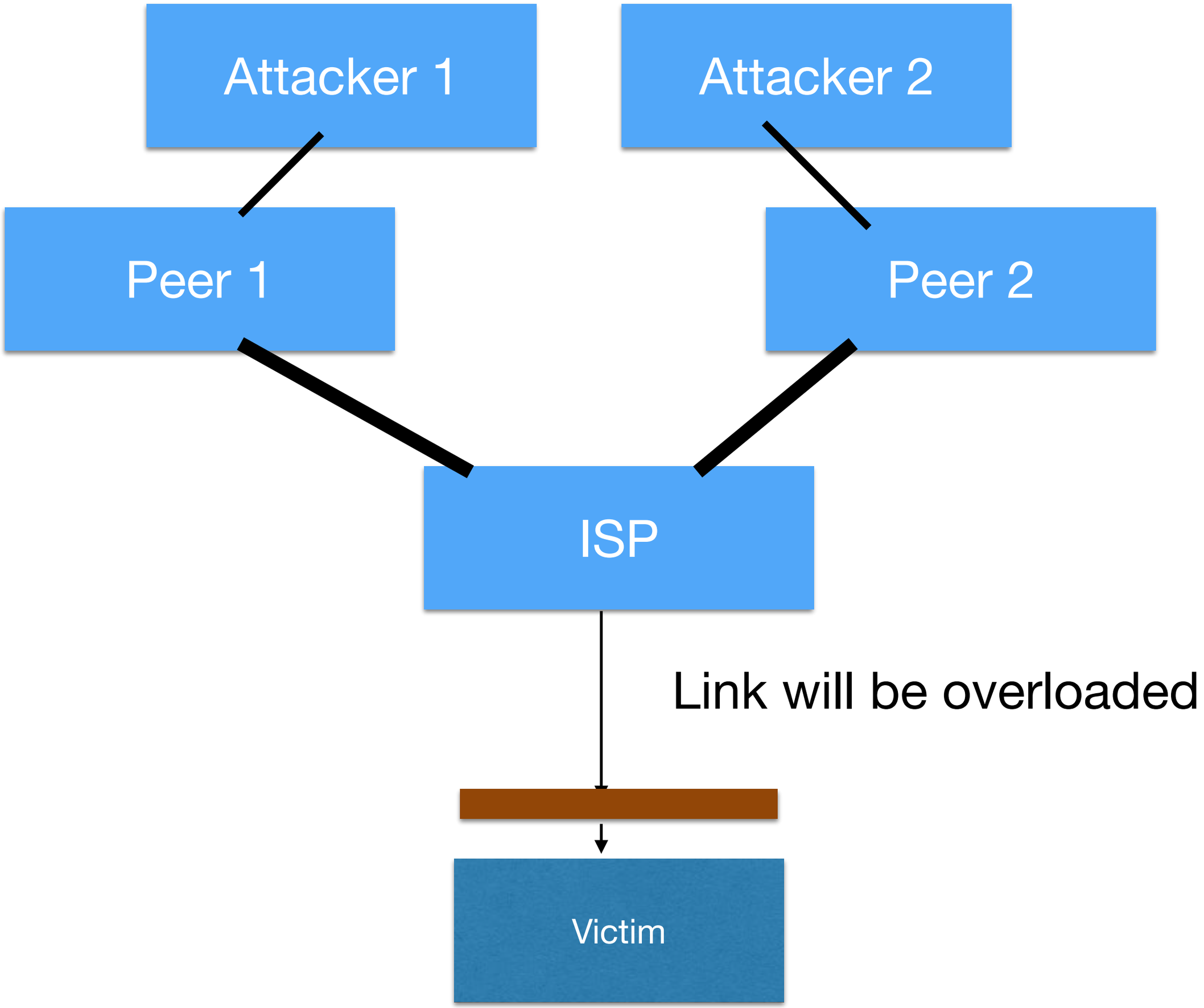
- Crude: lots of packets
- Less crude: lots of packets that do something (SYN, for example)
- Sophisticated: attacks that rely on knowledge of applications

Crude attacks

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

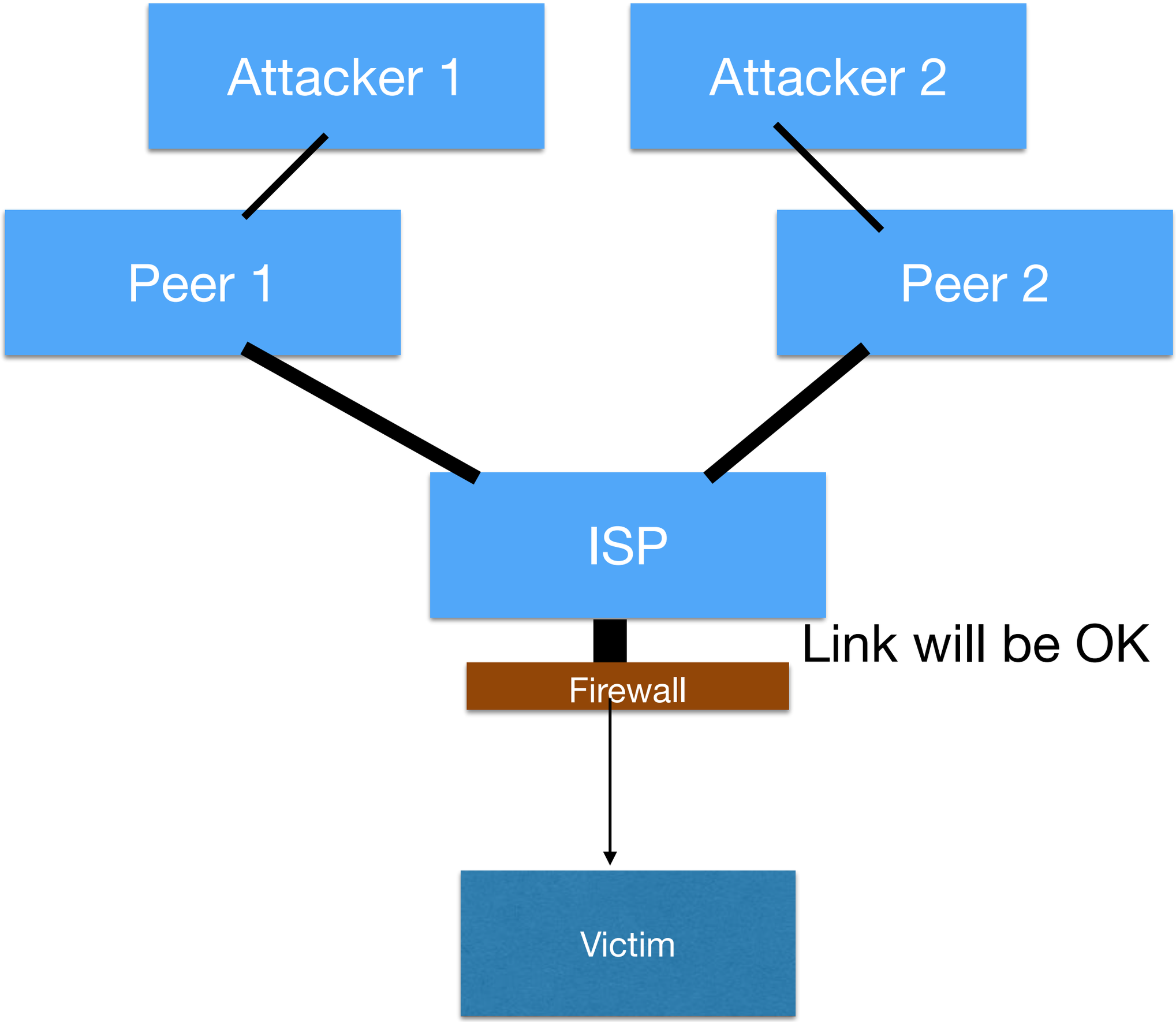
Crude Attacks

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


Attacker 1
Attacker 2
Peer 1
Peer 2
ISP
Link will be overloaded
Victim

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

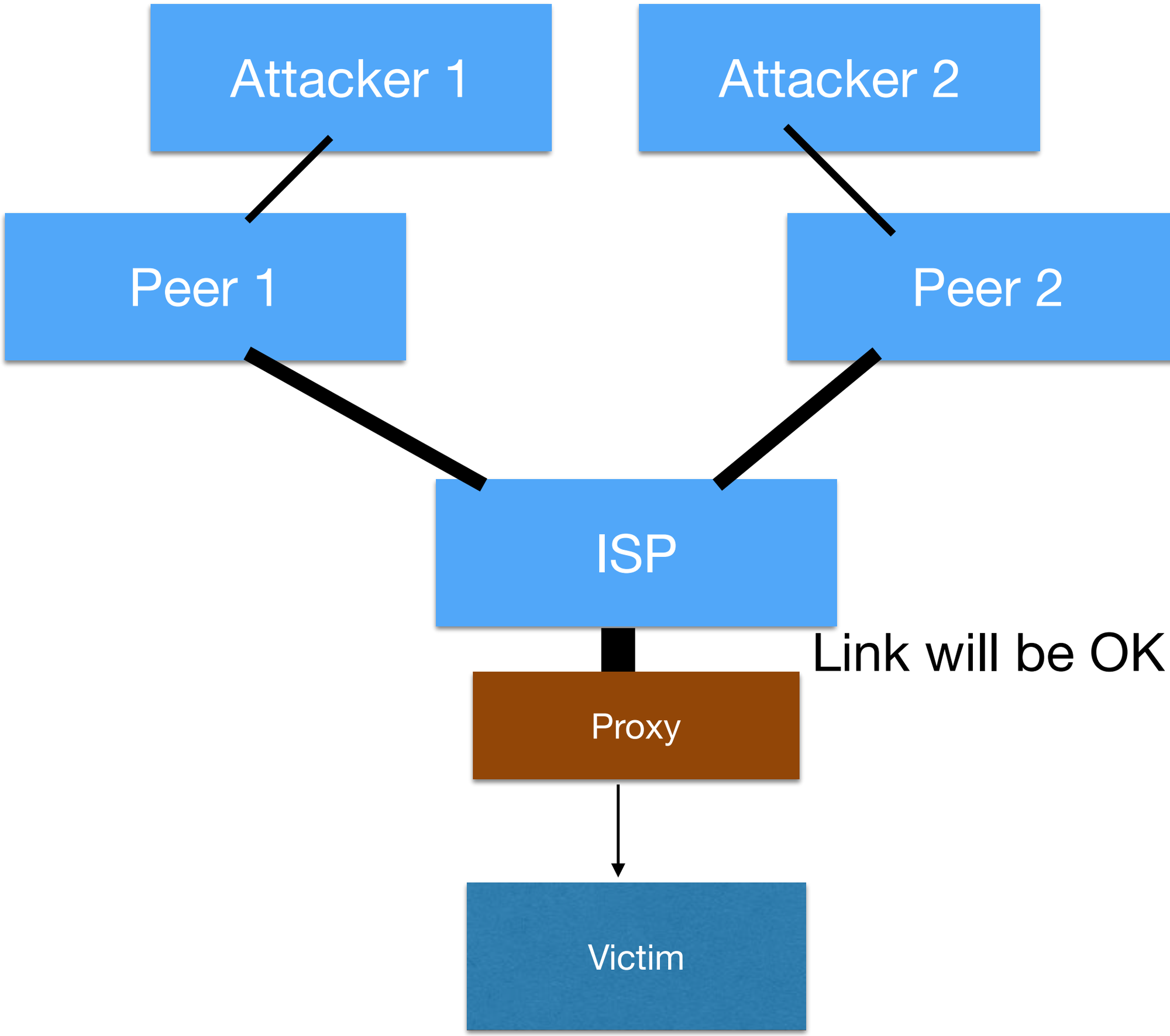

Attacker 1
Attacker 2
Peer 1
Peer 2
ISP
Firewall
Victim
Link will be OK

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


Attacker 1
Attacker 2
Peer 1
Peer 2
ISP
Proxy
Victim
Link will be OK

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 ≤ 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 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

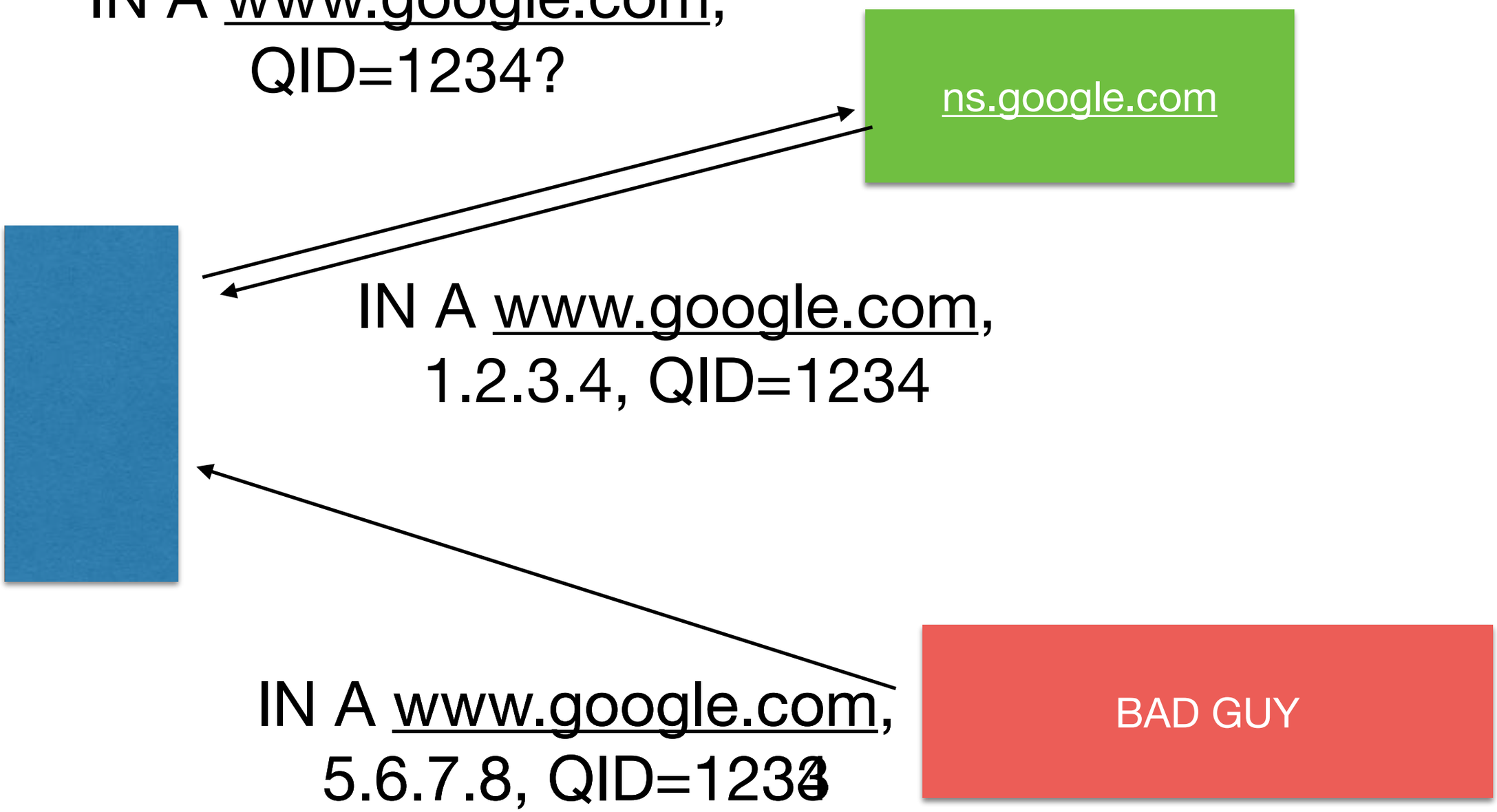
- Sometimes called Bogon or Martian Filtering
- Also worth filtering incoming and outgoing traffic that cannot be right
  - Example RFC1918 (10.0.0.0/8, etc) should never arrive and never leave, and should be dumped in both directions on all firewalls which touch the Internet
  - “How Ian stole email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

Kaminsky Attack

IN A www.google.com,
QID=1234?



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.obscuredom.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