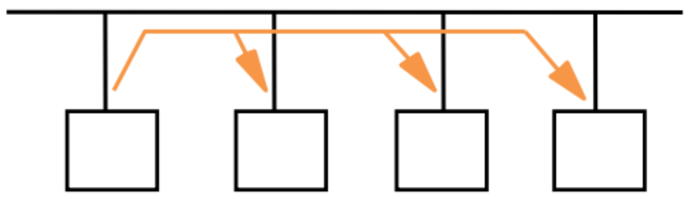
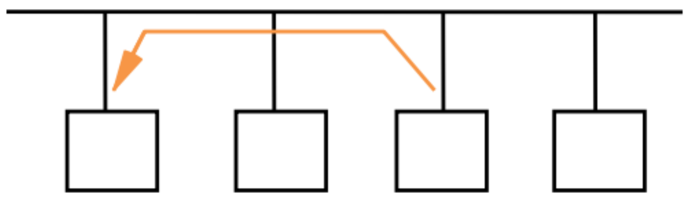
CS Revision Lecture 3, 4

- **Lecture 3 - Network Security: ARP, IP, TCP, UDP**
  - IP and MAC Addresses
    - Devices on a local area network have
      - **IP** addresses (network layer)
      - **MAC** addresses (data link layer)
    - IP addresses are used by high level protocols
    - MAC addresses are used by low level protocols
    - How to translate IP addresses into MAC addresses?
  - Address Resolution Protocol (ARP)
    - **Connects the network layer to the data link layer**
    - **Maps IP addresses to MAC addresses**
    - Based on broadcast messages and local caching
    - Does not support confidentiality, integrity, or authentication
    - Defined as a part of RFC 826(IETF, Request For Comments)
    - ARP Messages

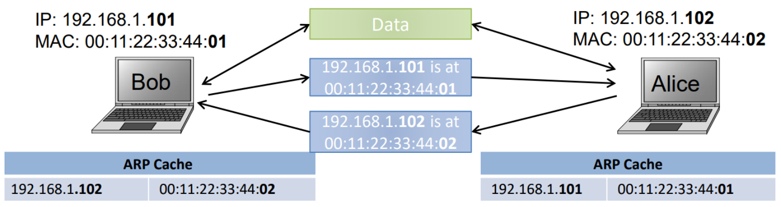




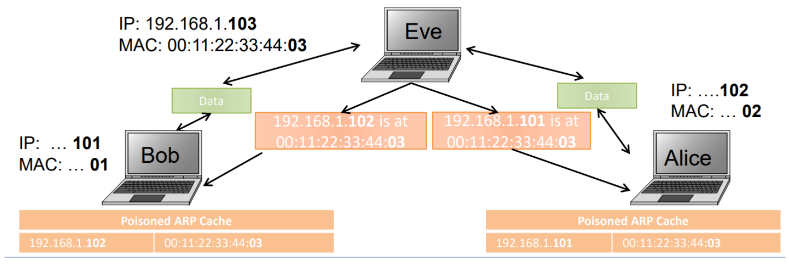
    - ARP **broadcasts** requests of type
        - **Who has** <IP addressC>
        - **tell** <IP addressA>
    - Machine with <IP addressC> responds <IP addressC> **is at** <MAC address>
    - Requesting machine caches response
    - Network administrator configures IP address and subnet on each machine
- ARP Cache
    - The Linux, Windows and OSX command **arp - a** displays the ARP table

```
Internet Address       Physical Address       Type
128.148.31.1           00-00-0c-07-ac-00      dynamic
128.148.31.15          00-0c-76-b2-d7-1d      dynamic
128.148.31.71          00-0c-76-b2-d0-d2      dynamic
128.148.31.75          00-0c-76-b2-d7-1d      dynamic
128.148.31.102         00-22-0c-a3-e4-00      dynamic
```

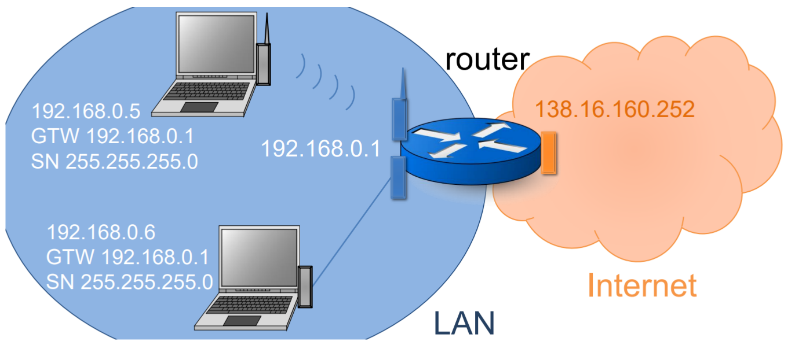
    - Command **arp -a -d** flushes the ARP cache (windows)
    - ARP cache entries are stored for a configurable amount of time
- ARP Cache Poisoning (aka ARP Spoofing)
    - The ARP table is updated whenever an ARP response is received
    - Requests are not tracked
    - ARP announcements are not authenticated
    - Machines trust each other
    - A rogue machine can spoof other machines
- ARP Normal Operation
    - Normal Operation
        - Alice communicates with Bob



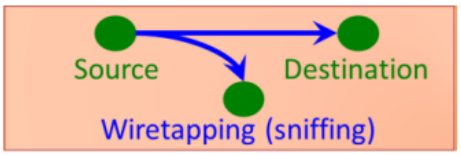
- ARP Cache Poisoning Attack
  - Mal actor-in-the-middle attack (MitM)
    - ARP cache poisoning leads to eavesdropping

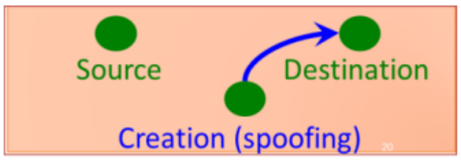


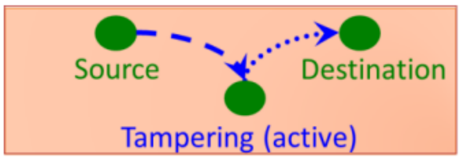
  - Almost all ARP implementations are stateless
  - An ARP cache updates every time that it receives an ARP reply
    - even if it did not send any ARP request
  - Can "poison" ARP cache with gratuitous ARP replies
    - A **Gratuitous ARP is an ARP Response that was not prompted by an ARP Request.**
  - Using static entries solves the problem but it is almost impossible to manage!
- From the LAN to the Internet

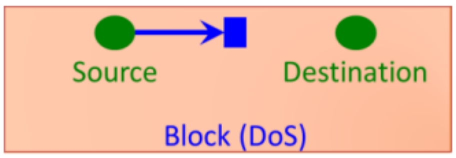


- IP Vulnerabilities







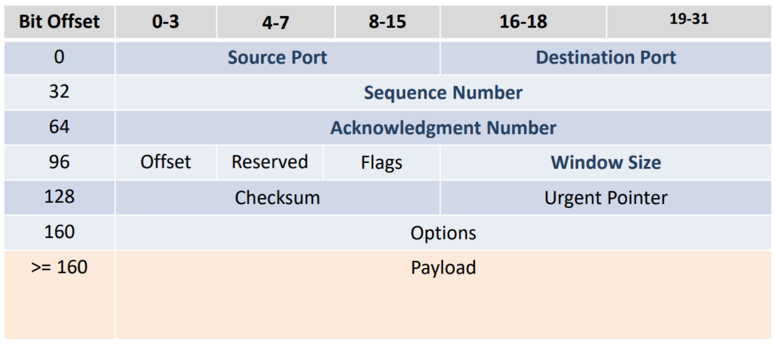


- Unencrypted transmission
- No Source authentication
  - Sender can **spoof source address**, making it difficult to trace packet back to attacker
- No integrity checking
  - Entire packet, header and payload, can be modified, enabling **content forgeries**, **redirections** and **mal actor- in-the-middle attacks**
- No bandwidth constraints
  - Large number of packets can be injected into network to launch a **denial-of-service attack**
  - Broadcast addresses provide additional leverage

- User Datagram Protocol (UDP)
  - UDP is a **stateless**, **unreliable** datagram protocol **built on top of IP**, i.e. it is at the **transport layer**
  - UDP does **not provide delivery guarantees or acknowledgements**, which makes it **efficient**

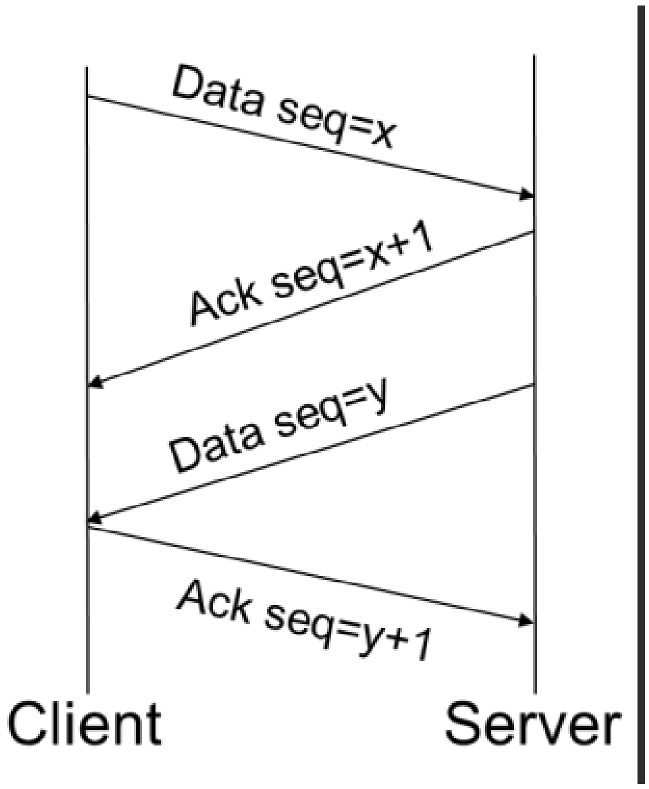
        - Can however distinguish data for **multiple concurrent applications** on a single host
        - A lack of reliability implies applications using UDP must be ready to accept a fair amount of corrupted and lost data
            - Most applications build on UDP will suffer if they require reliability
            - VoIP, streaming video and streaming audio all use UDP
    - Transmission Control Protocol (TCP)
        - Transport layer protocol for **reliable** data transfer, **in-order delivery** of messages and ability to distinguish **multiple applications** on same host
            - HTTP and SSH are built on top of TCP
        - **Stateful**: it keeps track of connection state in memory
        - TCP packages a data stream into segments transported by IP
            - Order maintained by marking each packet with a **sequence number**
            - Every time TCP receives a packet, it sends out an acknowledgement (ACK) to indicate successful receipt of the packet
        - TCP generally checks data transmitted by comparing a checksum of the data with a checksum encoded in the packet
    - Ports
        - TCP (& UDP) supports concurrent applications on the same server
        - Ports are **16 bit numbers** identifying where data is directed

        - telnet 192.168.0.1:80 https://example.co.uk:8080
    - The TCP **header** includes both a **source** and a **destination** port
    - Ports 0 through 1023 are reserved for use by know protocols
        - E.g., HTTPS uses 443 and SSH uses 22
    - Ports 1024 through 49151 are known as user ports, and are used for listening to connections
- TCP Packet Foramt



Bit Offset	0-3	4-7	8-15	16-18	19-31
0	Source Port			Destination Port	
32	Sequence Number				
64	Acknowledgment Number				
96	Offset	Reserved	Flags	Window Size	
128	Checksum			Urgent Pointer	
160	Options				
>= 160	Payload				

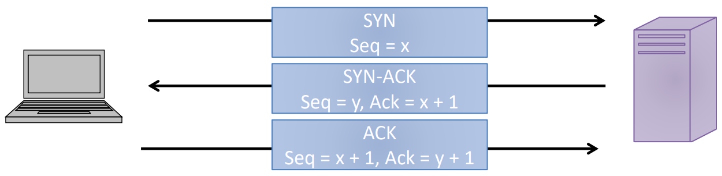
- TCP Data Transfer



    - During connection initialization using the three way handshake, **initial sequence numbers** are exchanged

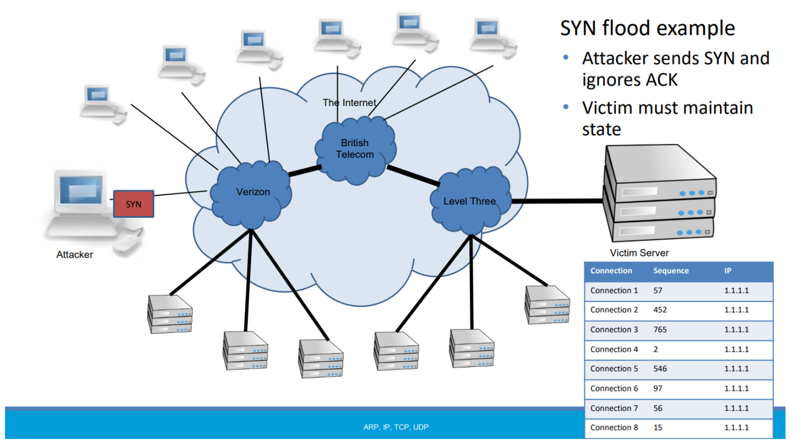
- The TCP header includes a **16 bit checksum** of the data and parts of the header, including the source and destination
- ACKs (or lack thereof) and window size are used by TCP to keep track of:
  - **packet loss**
  - **network congestion**
  - **flow control**

- Establishing TCP Connections



  - TCP connections are established through a **three-way handshake**.
  - The server generally is a passive listener, waiting for a connection request
  - The client requests a connection by sending out a SYN packet
  - The server responds by sending a SYN/ACK packet, acknowledging the connection
  - The client responds by sending an ACK to the server, thus establishing connection

- SYN Flooding
  - Send thousands of SYN requests to the victim
    - Alice sends many SYN packets, without acknowledging any replies. Bob accumulates more SYN packets than he can handle (i.e. runs out of space in state table).

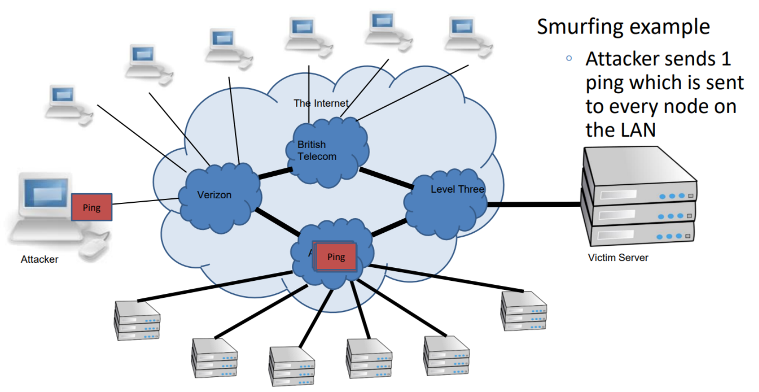


SYN flood example

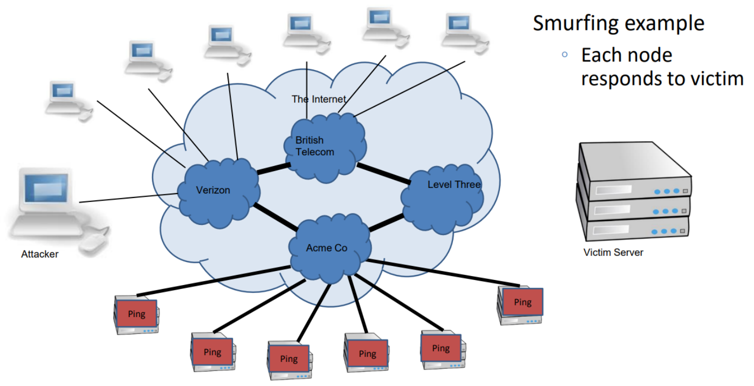
- Attacker sends SYN and ignores ACK
- Victim must maintain state

Connection	Sequence	IP
Connection 1	57	1.1.1.1
Connection 2	452	1.1.1.1
Connection 3	765	1.1.1.1
Connection 4	2	1.1.1.1
Connection 5	546	1.1.1.1
Connection 6	97	1.1.1.1
Connection 7	56	1.1.1.1
Connection 8	15	1.1.1.1

  - Problems
    - Attribution - attacker uses their own IP which could be traced
    - Bandwidth - attacker uses their own bandwidth which is likely smaller than a server's
  - Effective against a small target
    - Someone running a game server in their home
  - Not effective against a large target
    - Company website
- Spoofing: Forged TCP packets
  - Same as SYN flooding, but forget the source of the TCP packet
  - Advantages:
    - Harder to trace
    - ACKs are sent to a second computer, less attacker bandwidth used
  - Problems:
    - Ingress filtering is commonly used to drop packets with source addresses outside their origin network fragment
- Smurfing (directed broadcast)


Smurfing example

- Attacker sends 1 ping which is sent to every node on the LAN


Smurfing example

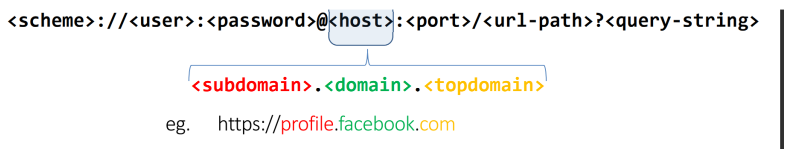
- Each node responds to victim

- The smurfing attack exploits ICMP (Internet Control Message Protocol) ping requests whereby remote hosts respond to echo packets to say they are online
- Some networks respond to pings to broadcast addresses. We call these networks “Smurf amplifiers”
- Idea: Ping a LAN on a broadcast address, then all hosts on the LAN reply to the sender of the ping
- Attack
  - Make a forged packet with the victim’s IP address as the source
  - Send it to a Smurf amplifier, which then causes a huge number of replies to the victim
- This is a form of reflection attack
- LANs that allow Smurf attacks are badly configured. One approach is to blacklist these LANs.

- Distributed Denial of Service (DDos)

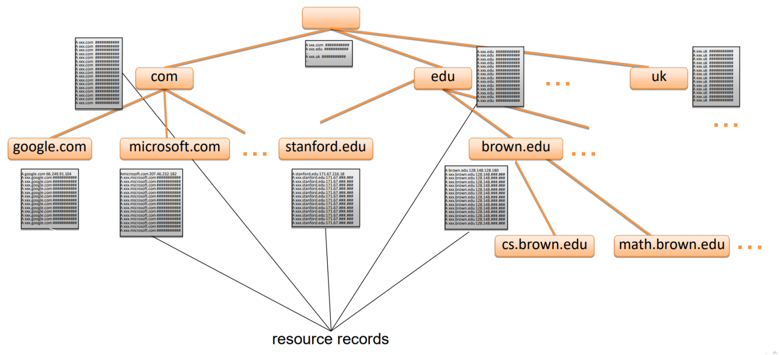
        - A large number of machines work together to perform an attack that prevents valid users from accessing a service.
    - What we have learned
        - ARP protocol
        - ARP poisoning attack
            - MitM attack on a LAN
        - Network and transport layer protocols
            - ICMP
            - TCP for reliable transmission
            - UDP when packet loss/corruption is tolerated
            - Dos Attacks: SYN flooding, Smurf
        - Lack of built-in security in network protocols
            - In future lectures we'll see how security must be incorporated at the application layer (e.g. TLS)
- **Lecture 4 - Network Security: Application-Layer and Domain Name System**
    - Sample Application-Layer Protocols
        - Domain name system (**DNS**)
        - Hypertext transfer protocol (**HTTP**)
        - **SSL/TLS**. Protocol used for secure, encrypted browsing (**HTTPS**)
        - **IMAP/POP/SMTP**. Internet email protocols
        - File transfer protocol (**FTP**). An old but still used protocol for uploading and downloading files
        - **Telnet**. Early remote access protocol
        - **SSH**. More recent secure remote access protocol
    - What is a URL?

    - Uniform Resource Locators (URLs) are a standardized format for describing the location and access method of resources via the internet


<scheme>://<user>:<password>@<host>:<port>/<url-path>?<query-string>
<subdomain>.<domain>.<topdomain>
eg. https://profile.facebook.com

- Domain name System
    - The **domain name system (DNS)** is an application-layer protocol
    - Basic function of DNS
        - Map domain names to IP addresses
        - The mapping is many to many
    - Examples:
        - www.ed.ac.uk and edwc.is.ed.ac.uk map to 129.215.228.101
        - google.com maps to 216.58.213.110, 198.7.237.249, and other addresses
    - More generally, DNS is a distributed database that stores **resource records**
        - **Address** (A) record: IP address associated with a host name
        - **Mail exchange** (MX) record: mail server of a domain
        - **Name server** (NS) record: authoritative server for a domain
- Domains
    - Domain name
        - Two or more labels, separated by dots (e.g., inf.ed.ac.uk)
    - Top-level domain (TLD)
        - Generic (gTLD), e.g., **.com, .org, .net**

    - Country-code (ccTLD), e.g., **.ca, .it**
    - New top level domains, e.g., **.scot, .tirol**
  - ICANN (*Internet Corporation for Assigned Names and Numbers)*
    - (non-profit) Internet Corporation for Assigned Names and Numbers
    - Keeps database of registered gTLDs (InterNIC)
    - Accredits registrars for gTLDs
  - gTLDs (Generic top-level domain)
    - Managed by ICANN
  - ccTLDs (Country code top-level domain)
    - Managed by government organizations
  - DNS Tree



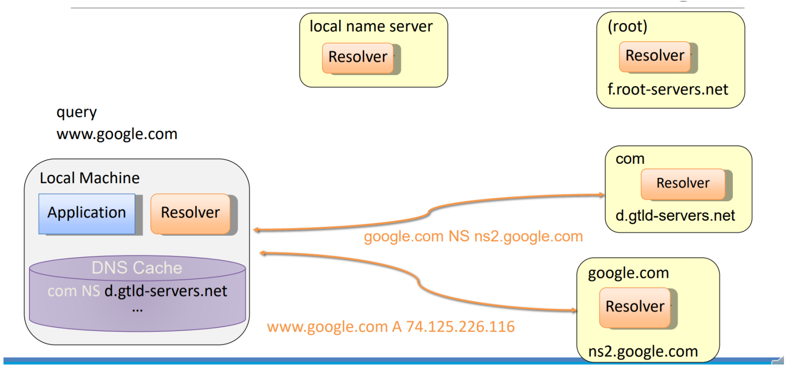
- Name Servers
  - Name server
    - Keeps local database of DNS records
    - Answers DNS queries
    - Can ask other name servers if record not in local database
  - Authoritative name server
    - Stores reference version of DNS records for a zone (partial tree)
  - Examples

    - dns0.ed.ac.uk is authoritative for ed.ac.uk and dns0.inf.ed.ac.uk for inf.ed.ac.uk
  - Root servers
    - Authoritative for the root zone (TLDs)
    - [a-m].root-servers.net
    - Supervised by ICANN
- Name Resolution
  - Resolver
    - Program that retrieves DNS records
    - Connects to a name server (default, root, or given)
    - E.g., **dig** in Linux and **nslookup** in Windows
    - Caches records received
  - **Iterative resolution**
    - Name server refers client to authoritative server (e.g., a TLD server) via an NS record
    - If the server is an authority for the name, it sends the answer. if it is not, it returns (to the client) the IP address of the server that it thinks can resolve the query. The client is responsible for repeating the query to this second server. If the newly addressed server can resolve the problem, it answers the query with the IP address, Otherwise it returns the IP address of a new server to the client. Now the client must repeat the query to the third server. This process is called iterative because the client repeats the same query to multiple servers.
  - **Recursive resolution**
    - Name server queries another server and forwards the final answer (e.g., A record) to client

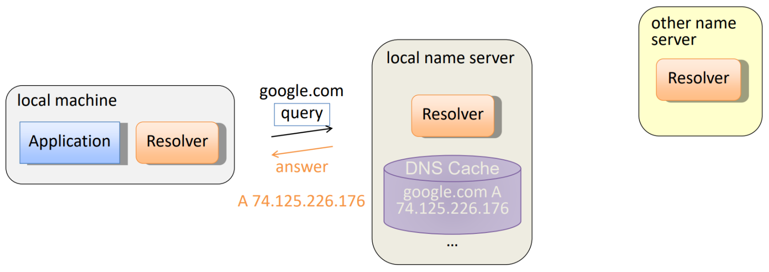
        - This means that the resolver expects the server to supply the final answer. If the server is the authority for the domain name, it checks its database and responds. If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response. If the parent is the authority, it responds, otherwise, it sends the query to yet another server. When the query is finally resolved the response travels back until it finally reaches the requesting client.
- Glue Records
    - Circular references
        - The authoritative name server for a domain may be within the same domain
        - E.g., dns0.inf.ed.ac.uk is authoritative for inf.ed.ac.uk
    - Glue record
        - Record of type A (IP address) for a name server referred to NS record Essential to break circular references
        - E.g.

            > inf.ed.ac.uk. NS dns0.inf.ed.ac.uk.
            > dns0.inf.ed.ac.uk. A 129.215.160.240 [glue record]
- DNS Caching
    - There would be too much network traffic if a path in the DNS tree would be traversed for each query
        - Root servers and TLD servers would be rapidly overloaded
    - DNS servers **cache** records that are results of queries for a specified amount of time
        - Time-to-live field
    - DNS queries with caching

        - First, resolver looks in cache for A record of query domain
        - Next , resolver looks in cache for NS record of longest suffix of query domain
- Iterative Name Resolution with Caching



- Recursive Name Resolution with Caching



- Local DNS Cache
    - Operating system maintains DNS cache
        - Shared among all running applications
        - Can be displayed to all users
        - View DNS cache in Windows with command **ipconfig /displaydns**
        - Clear DNS cache in Windows with command **ipconfig /flushdns**
- DNS Cache Poisoning
    - Basic idea
        - Give a DNS server a false address record and get it cached

    - DNS query mechanism
        - Queries issued over UDP on port 53
        - 16-bit request identifier in payload to match answers with queries
        - No authentication
    - Cache may be poisoned when a resolver
        - Query has predictable identifiers and return ports
        - Attacker answers before authoritative name server
        - Ignore identifier, accepts unsolicited DNS records
    - Early versions of BIND (popular DNS software) vulnerable to cache poisoning
    - How to defense
        - Query randomization
            - **Random request identifier** (16 bits)
            - **Random return port** (16 bits)
        - Probability of guessing request ID or return port
            - $1/2^{16} = 0.0015\%$
        - Probability of guessing request ID and return port is
            - $1/2^{32}$ (less than one in four billion)
        - Birthday Paradox
- Subdomain DNS Cache Poisoning (Kaminsky)
    - Attacker causes victim to send
        - Many DNS requests for nonexistent subdomains of target domain
    - Attacker sends victim
        - Forged NS responses for the requests
    - Format of forged response
        - Random ID

        - Correct NS recor
        - Spoofed glue record pointing to the attacker’s name server IP
- DNSSEC (Domain Name System Security Extensions)
    - Goals
        - Authenticity of DNS answer origin
        - Integrity of reply
        - Authenticity of denial of existence
    - Implementation
        - Signed DNS replies at each step
        - Public-key cryptography
    - Slow deployment
        - Root servers support since 2010
- What We Have Learned
    - How DNS operates
        - Distributed database
        - Resolvers and name servers
        - Iterative vs. recursive resolution
        - Caching
    - DNS cache poisoning attacks
    - DNSSEC