UNIVERSITÀ DEGLI STUDI
DI TRENTO

OSI Session / presentation / application Layer

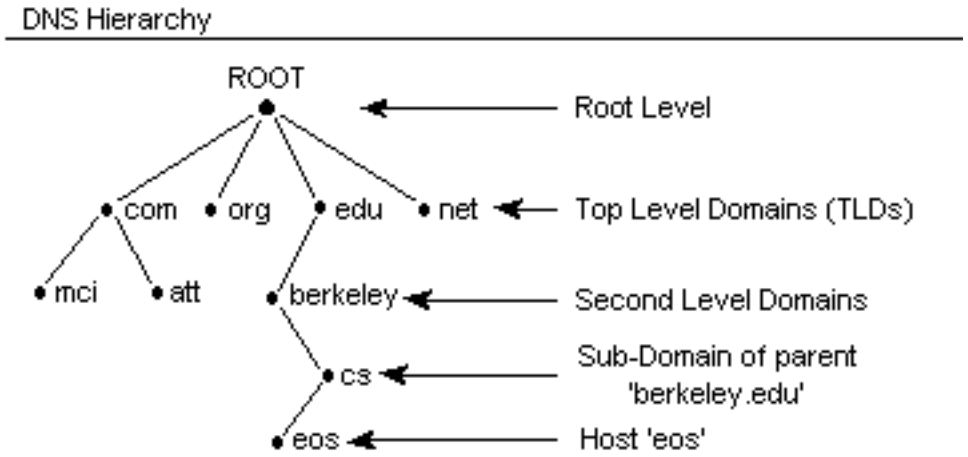
Higher level protocols

- On top of IP, TCP, UDP, etc. there are a plethora of application-level protocols
  - FTP → file transfer
  - SMTP/POP/IMAP → mail
  - Telnet → remote access
  - SSH → remote access
  - HTTP → web
  - DNS → infrastructure
  - …
- Pointless exercise to go through them all
- Rather, we focus on some most important threats

Domain Name Service (quick intro)

- DNS is a hierarchical system for domain name resolving
  - Translates human-readable addressed (google.com) to (a set of) IP addresses the domain is reachable at
  - UDP for fast answers (port 53)
- Each transaction identified by an ID (16 bits)
  - Transaction ID: "TXID"
  - Original DNS implementation → incremental TXID
- Several type of records. Of interest here
  - A (AAAA) → IPv4 (IPv6) of the requested domain
    - e.g. a.website.com A 65.61.198.201
  - NS → IP of the DNS server to ask
    - e.g. a.website.com NS ns.website.com
    - Followed by an A answer for the dns
      - ns.website.com A 2.2.2.2

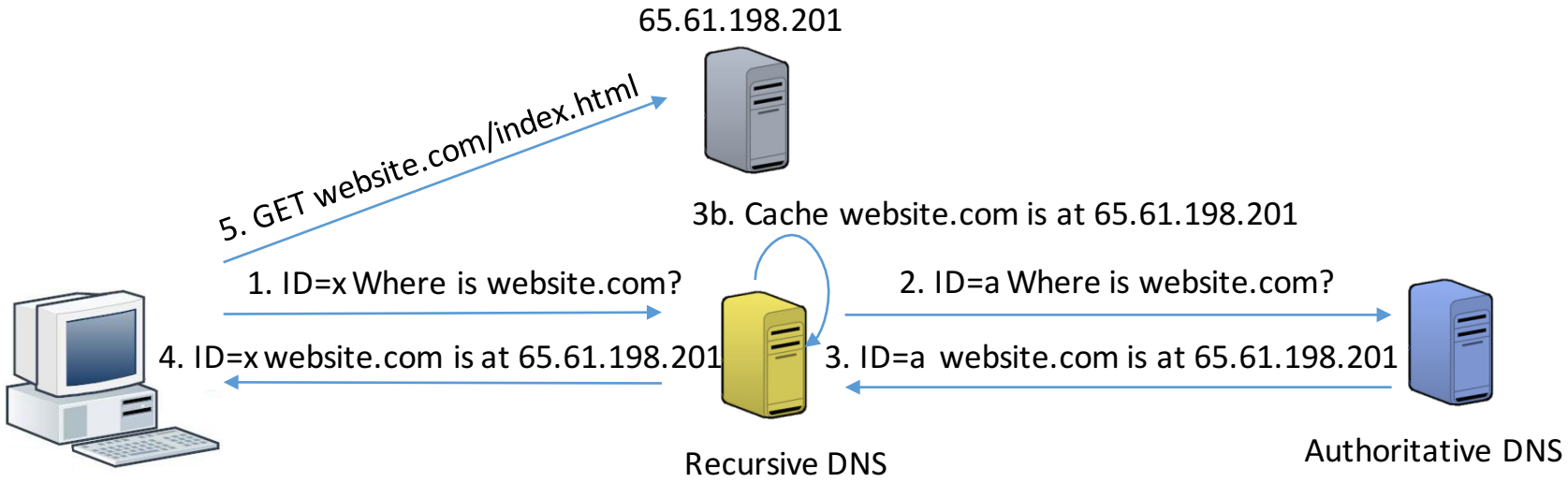
DNS hierarchy



- Root DNSs → responsible for top level domain queries
  - e.g. .com NS ns.auth.net
- Authoritative DNS → a DNS server that answers queries whose answer it already knows
  - Does not ask to other DNSs
- Recursive DNS → a DNS server that forwards queries to Authoritative DNSs

DNS queries, authoritative answers, and caching (simplified)

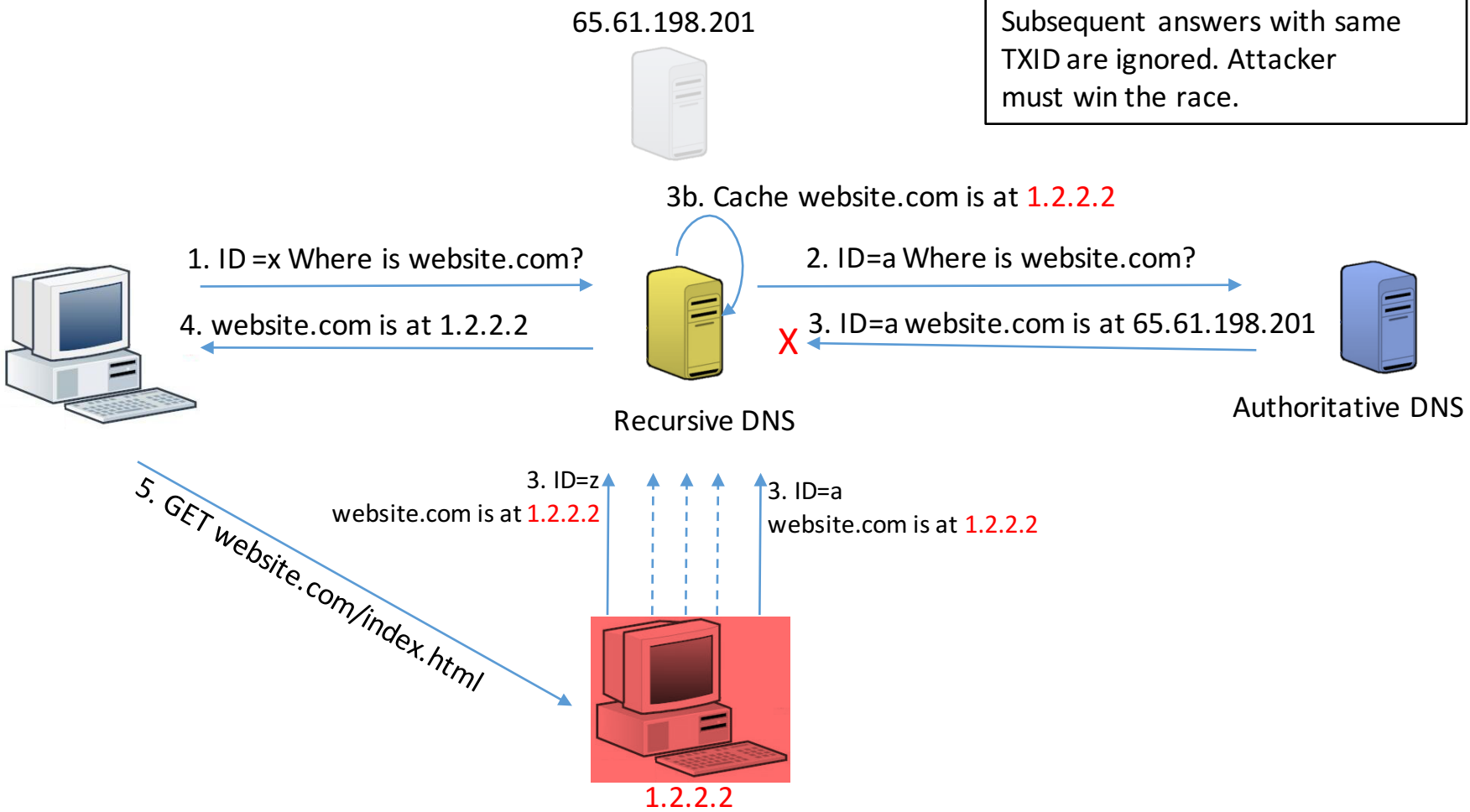
- When the client wants to contact www.website.com it sends a query to its local DNS (also called recursive DNS)
- Local DNS forwards request to authoritative DNS
- Local DNS caches entry



DNS cache poisoning

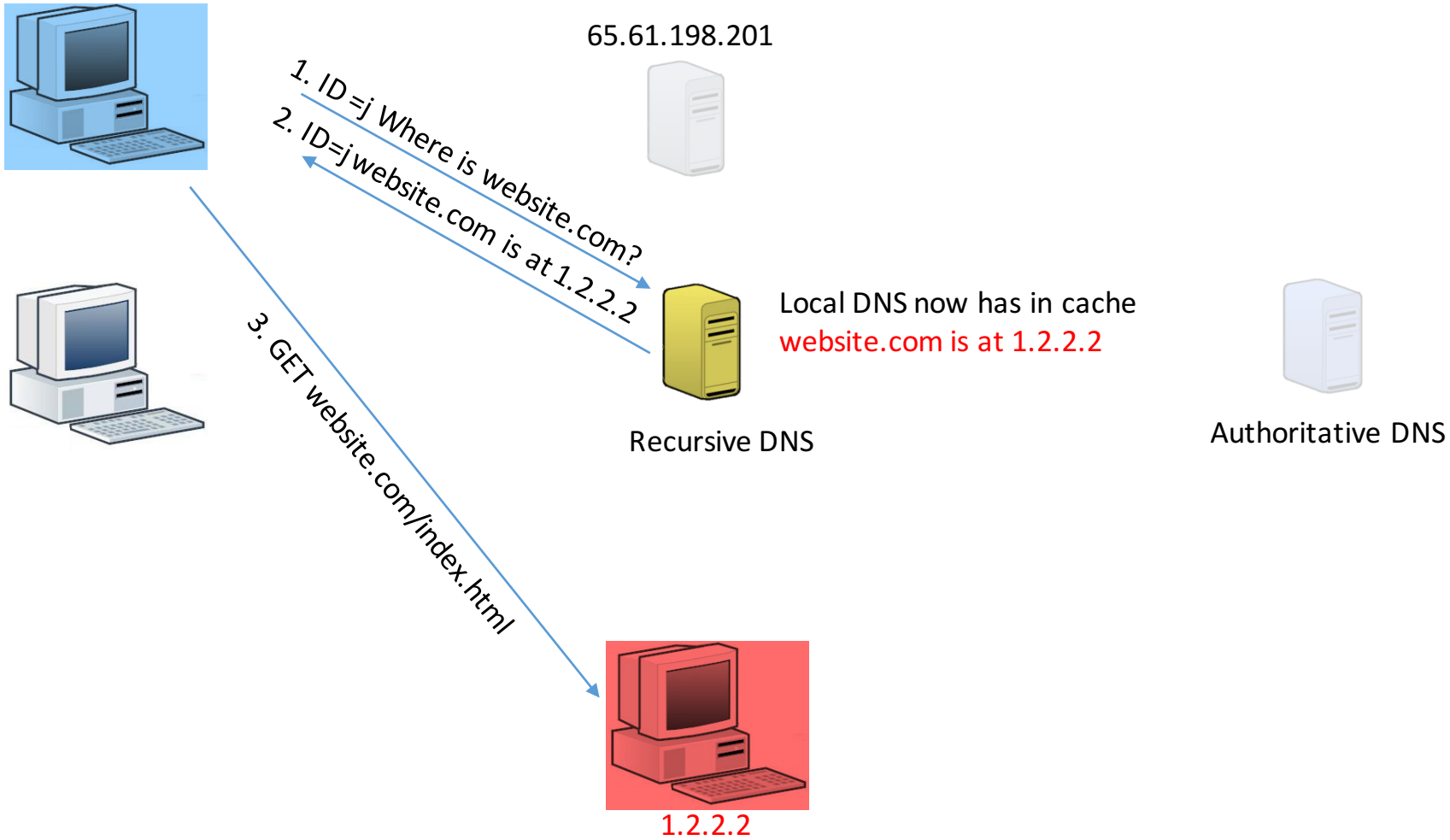
Recursive DNS' cache:
website.com A 1.2.2.2

The first received answer is cached
Subsequent answers with same TXID are ignored. Attacker must win the race.



DNS cache poisoning

Recursive DNS' cache:
website.com A 1.2.2.2



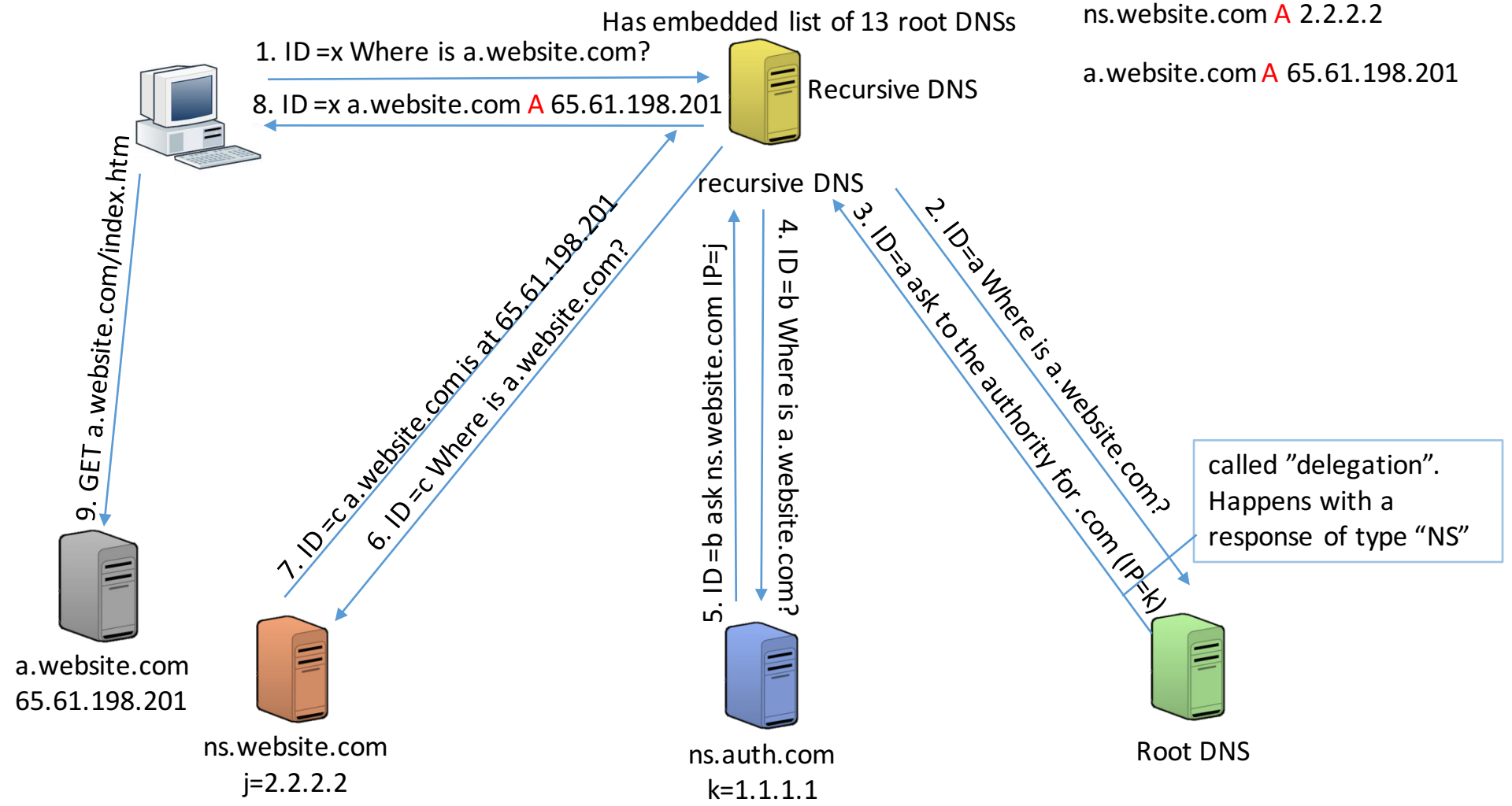
DNS, the full picture

Recursive DNS' cache:

.com NS ns.auth.com
ns.auth.com A 1.1.1.1

website.com NS ns.website.com
ns.website.com A 2.2.2.2

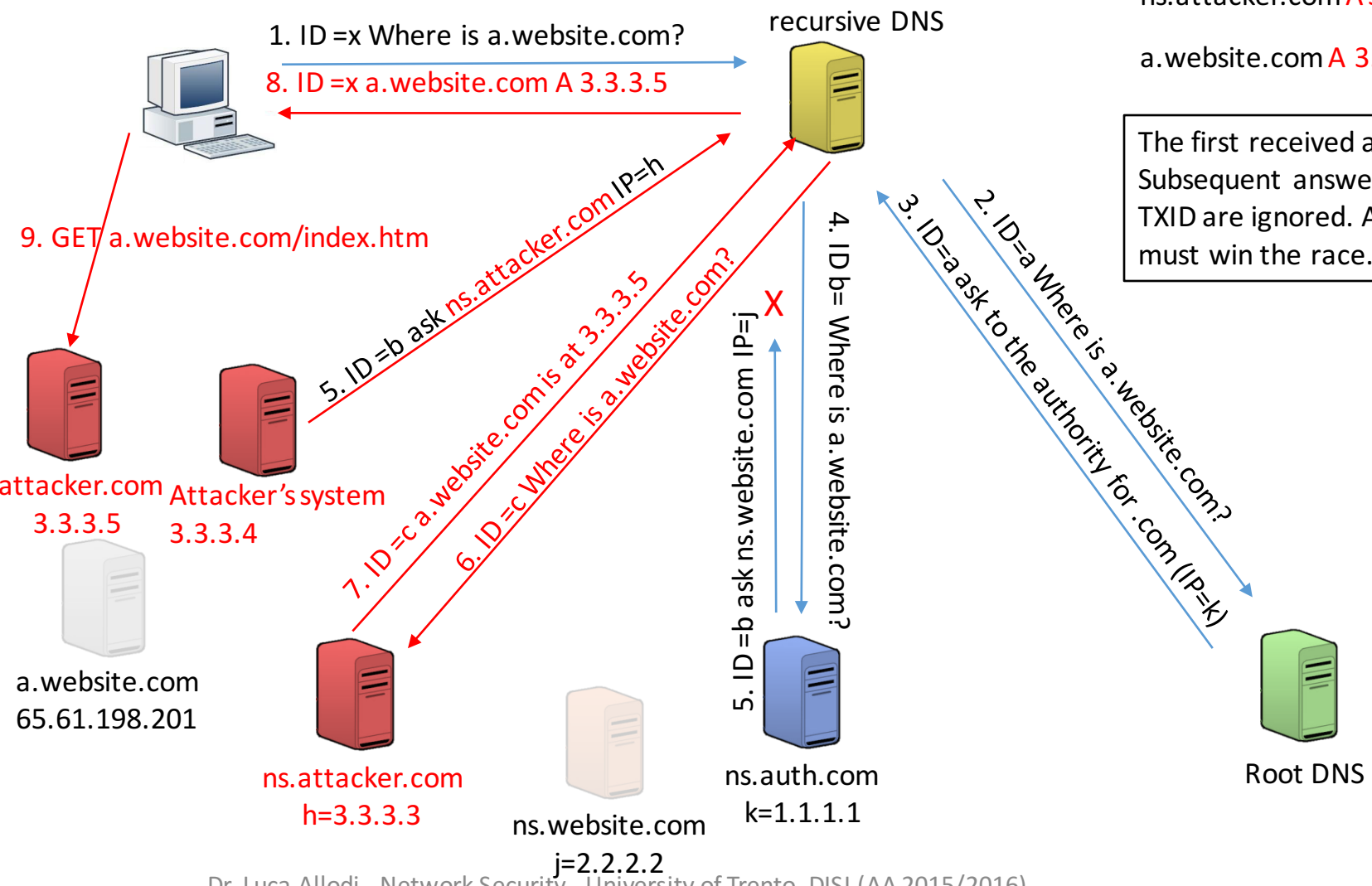
a.website.com A 65.61.198.201



Kaminsky vulnerability

- The Kaminsky vulnerability can lead to a cache poisoning attack
- The attacker rather than replacing an A record replaces an NS record
- This way the attacker can get control over any (sub)domain
  - b.a.website.com
  - a.website.com
  - website.com
  - .com

Kaminsky attack (cntd)



Recursive DNS' cache:

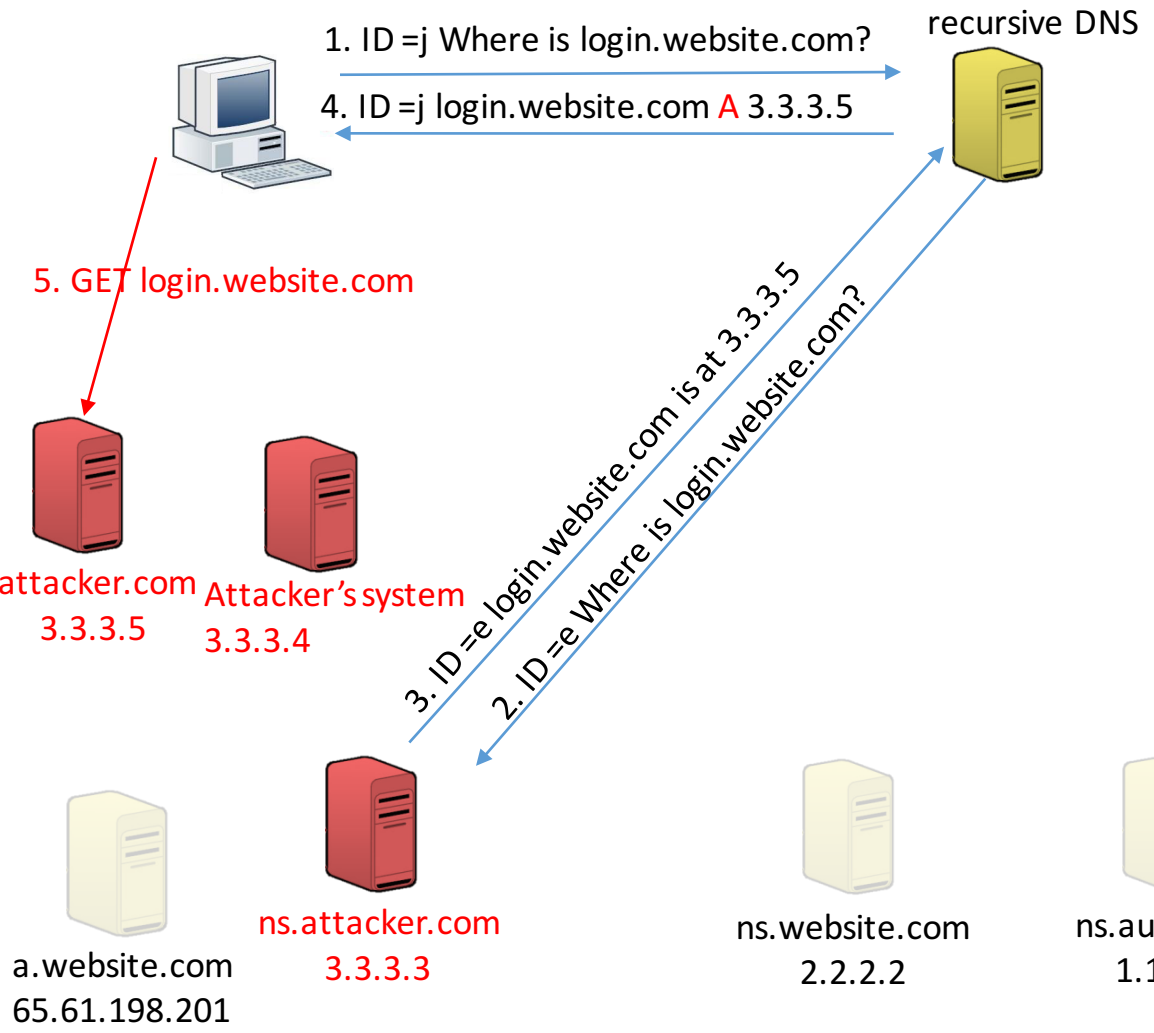
.com NS ns.auth.com
ns.auth.com A 1.1.1.1

website.com NS ns.attacker.com
ns.attacker.com A 3.3.3.3

a.website.com A 3.3.3.5

The first received answer is cached
Subsequent answers with same TXID are ignored. Attacker must win the race.

Kaminsky attack (cntd)



Recursive DNS' cache:

.com NS ns.auth.com
ns.auth.com A 1.1.1.1

website.com NS ns.attacker.com
ns.attacker.com A 3.3.3.3

a.website.com A 3.3.3.5

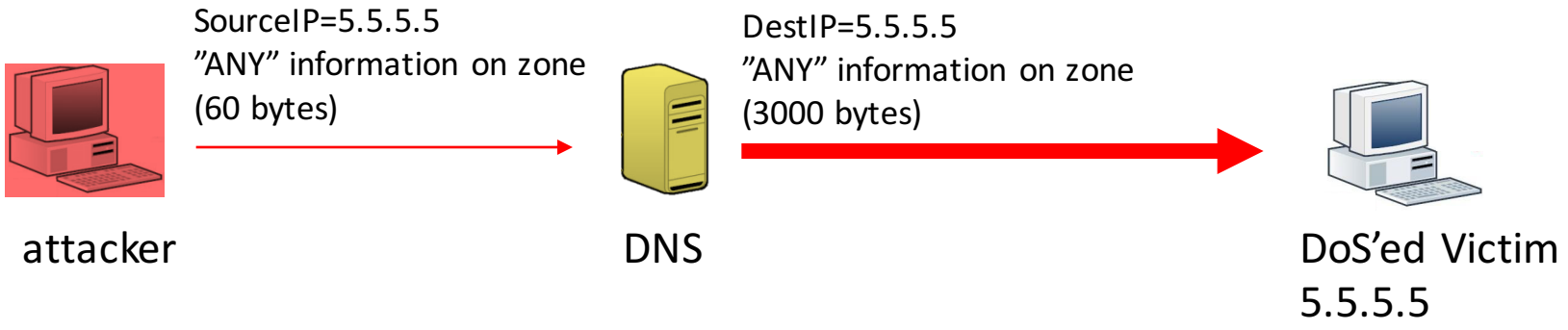
login.website.com A 3.3.3.5

Mitigation of Kaminsky's vulnerability

- Source of attack is low entropy with a 16 bit ID
  - Randomness is not enough to represent a significant margin
  - Moving ID size to 32 bits is not feasible
    - Can not change the protocol
- Solution → randomize the source port (16 bits) to increase entropy
  - In reality can't use all 16 bits for the source port because of reserved values
  - Any answer that does not match **both** source port and transaction ID will be dropped

DNS amplification attack

- A type of DoS attack
- Exploits certain type of DNS answers that are much bigger in size than the requests
  - attack's throughput much bigger than attacker's input
- DNS works over UDP → source IP easy to spoof



DNS zone transfer

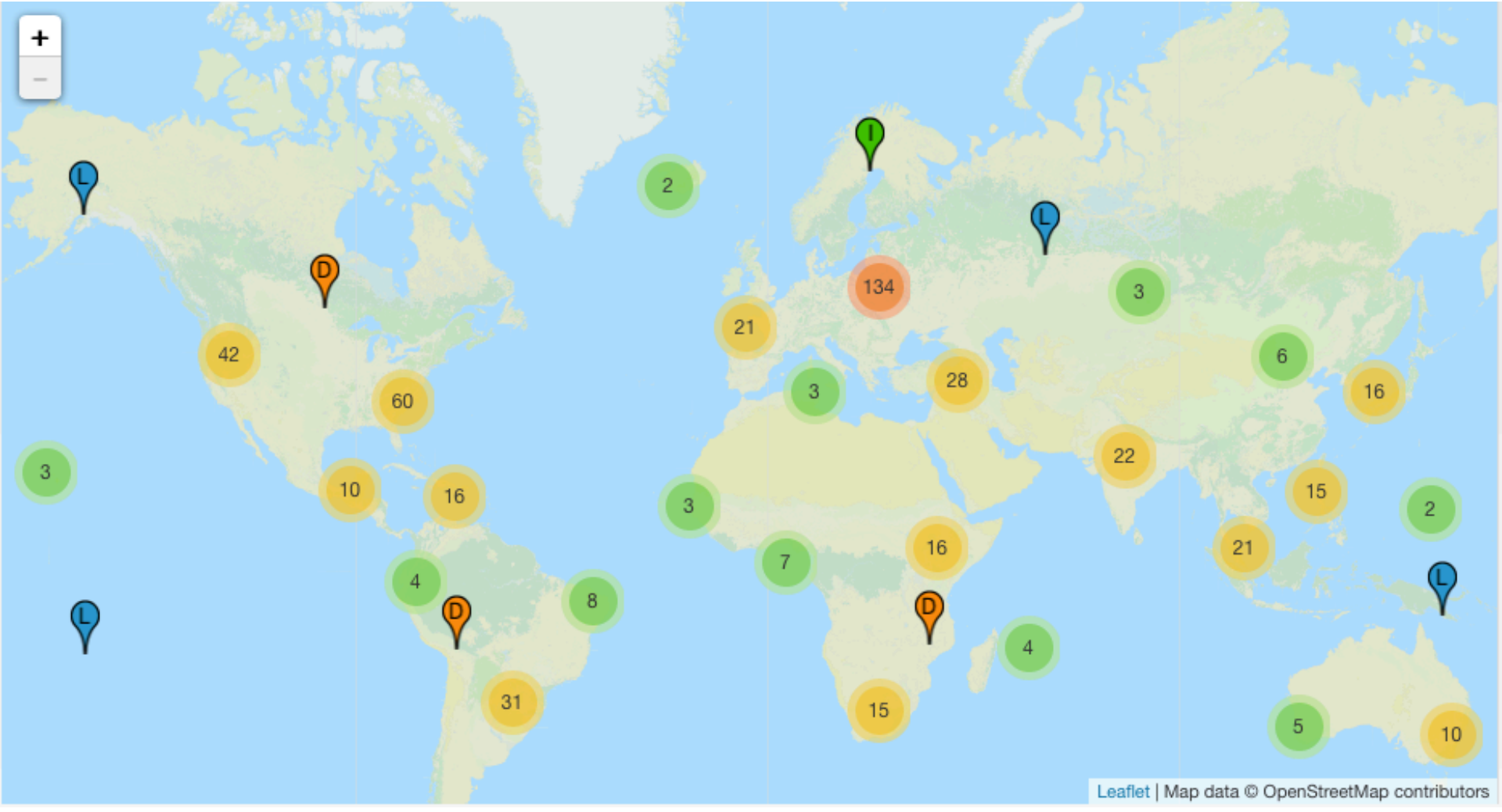
- A zone is a domain for which a server is authoritative
- "slave" servers can ask "authoritative" servers to copy their zone database
  - Over TCP
- An attacker pretends to be a slave server and dump the zone DB
  - Acquires knowledge of zone's infrastructure
  - Can be used to facilitate further attacks (e.g. spoofing or more direct attacks)

DNSSec

- Secure implementation of the DNS protocol
- Implements DNS authentication on top of normal DNS exchange
  - Digitally signed over a *chain-of-trust* starting from the root server
  - Uses electronic certificates
    - Public-key crypto → authenticate by showing proof that you own a secret key
- Protects data integrity
  - No confidentiality protection
- Additional reading
  - Hao Yang ; Osterweil, E. ; Massey, D. ; Songwu Lu ; Lixia Zhang. Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC. *IEEE Transactions on Dependable and Secure Computing. Vol 8, Issue 5.*

DNS root servers location

http://root-servers.org

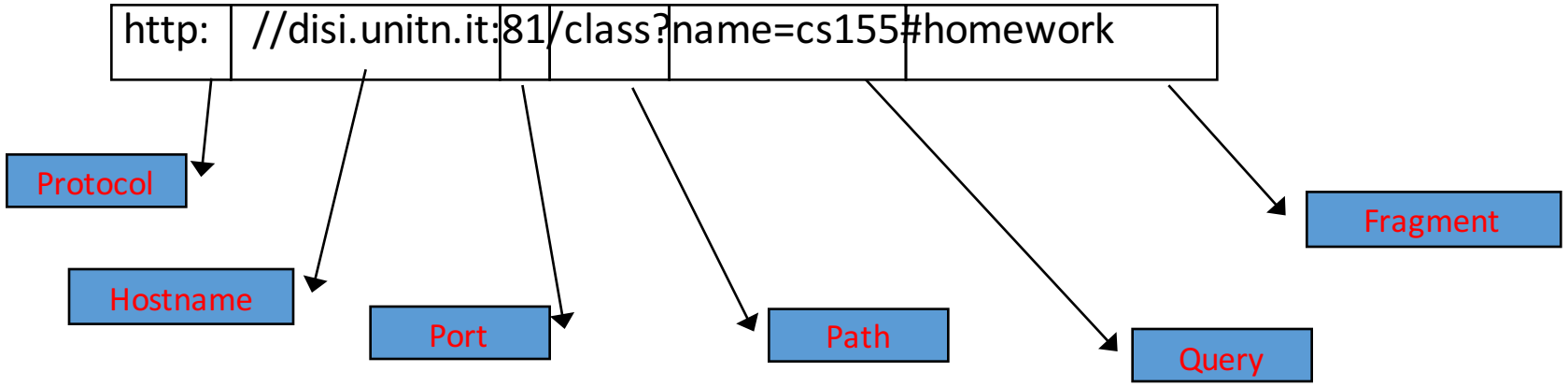


HTTP

- Main protocol on which the www works
- Based on the notion that client can either request or submit data to a server
- Two methods
  - GET → Requests data from a specified resource
    - GET /test/demo_form.asp**?name1=value1&name2=value2** HTTP/1.1
  - POST → Submits data to be processed to a specified resource
    - POST /test/demo_form.asp HTTP/1.1
      Host: w3schools.com
      name1=value1&name2=value2
- HTTP is Stateless
  - HTTP cookies enable statefulness

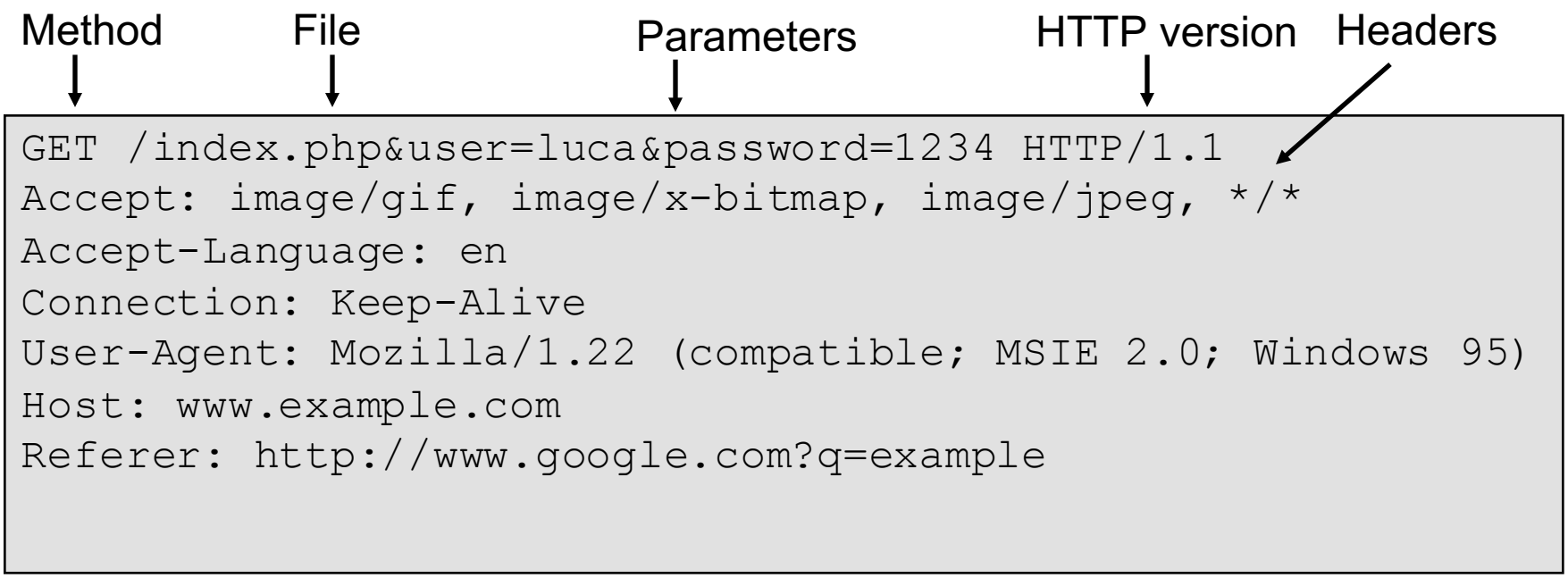
URLs

- Global identifiers of network-retrievable documents
- **Example:**



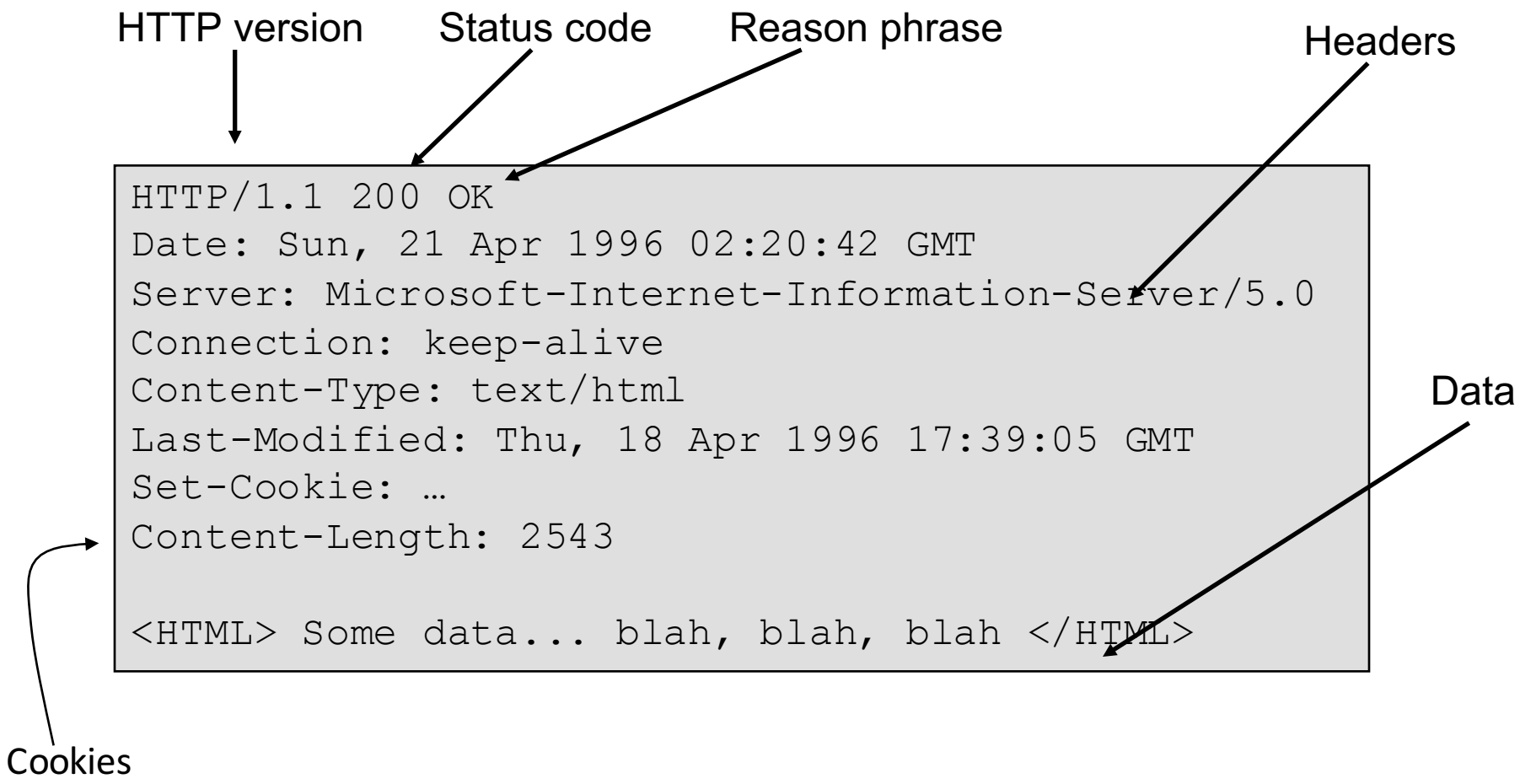
- Special characters are encoded as hex:
  - %0A = newline
  - %20 or + = space, %2B = + (special exception)

HTTP GET Request



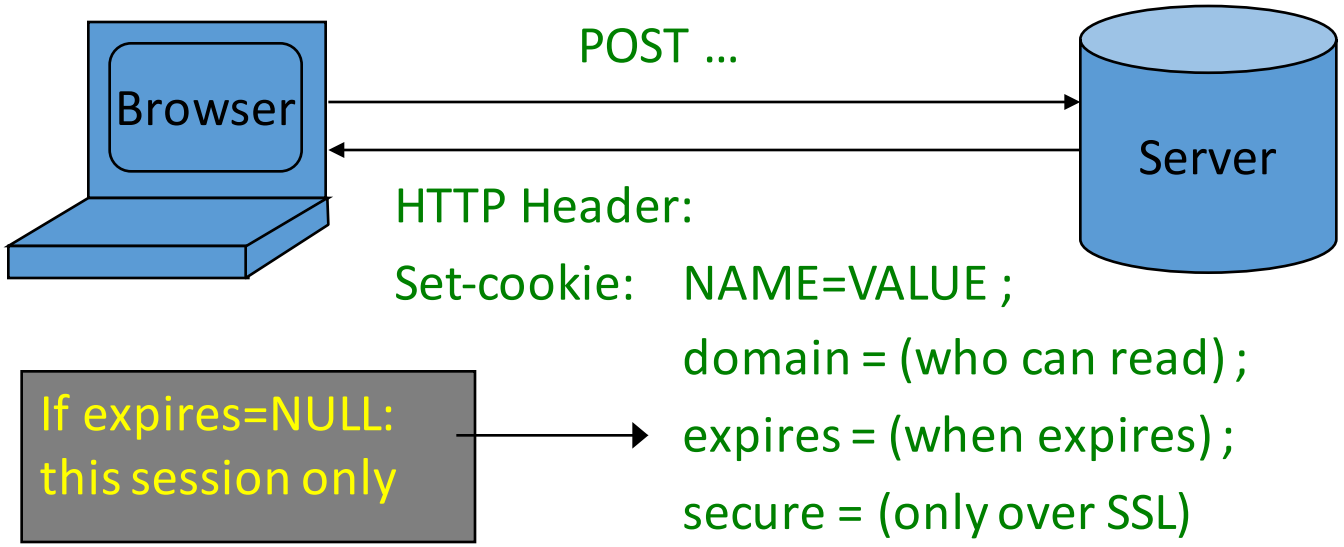
HTTP POST Request

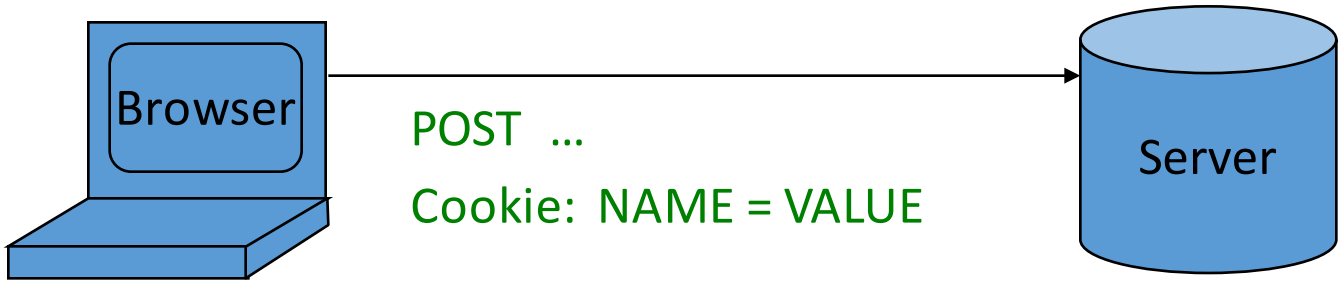
HTTP Response



Cookies

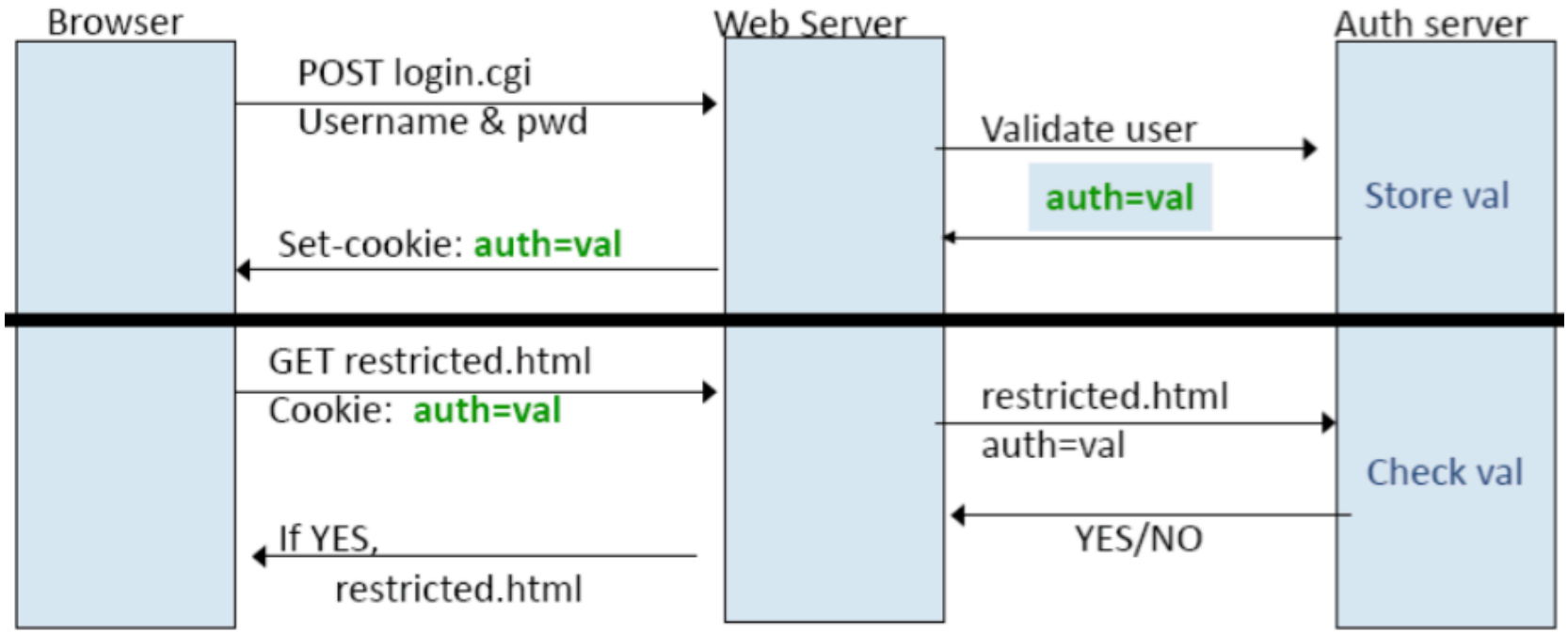
- Used to store state on user's machine



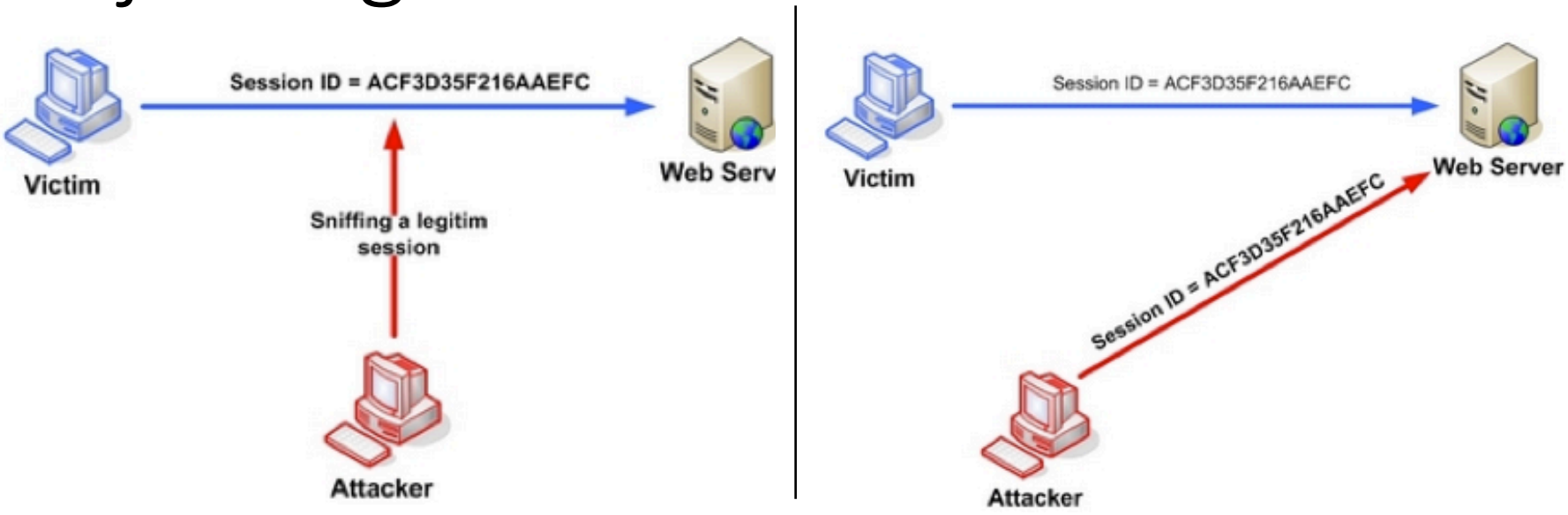


HTTP is stateless protocol; cookies add state

Cookie example: authentication

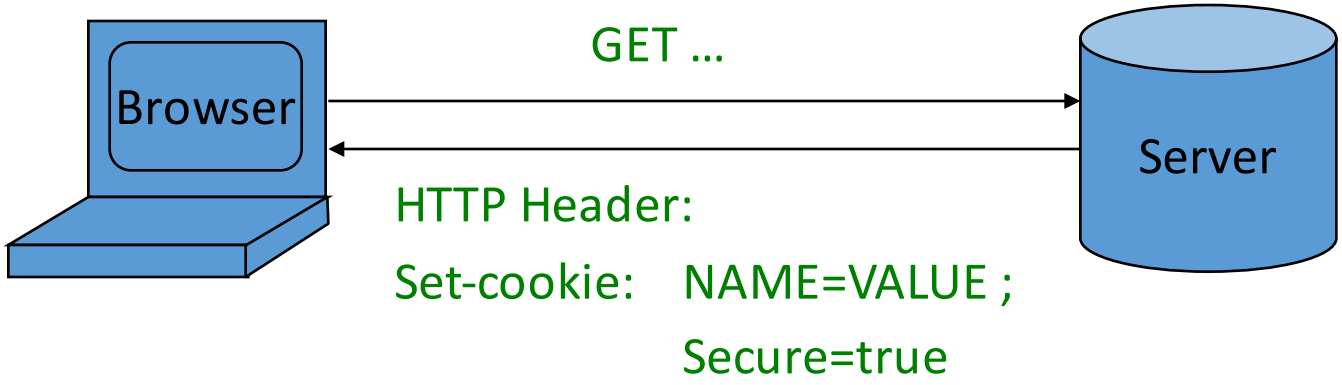


Attack example: HTTP session hijacking



- Session ID used by webserver to authenticate client "victim"
  - Send over cookie in-the-clear
- Attacker can read the session ID cookie and spoof the victim's identity
  - e.g. access to personal webpages/accounts (e.g. Facebook until 2011)
- https://www.owasp.org/index.php/Session_hijacking_attack

Secure Cookies



- Provides confidentiality against network attacker
  - Browser will only send cookie back over encrypted channels
- … but no integrity
  - Can rewrite secure cookies over HTTP

    ⇒⇒ network attacker can rewrite secure cookies

Telnet

- Protocol used in remote control services
  - Implemented through a virtual terminal that connects to systems
- Operates over TCP port 23
- Remote client can issue commands to server
  - Plain-text commands
  - Typically no authentication
  - Typically no channel encryption
- No need to go through possible attacks here
- Use SSH instead :-)

Common issues

- Most of the network attacks we've seen so far have at least one of two issues common among most network problems
  - Lack of authentication → the real sender/receiver of a packet/datagram can not be authenticated
    - It is possible to spoof its identity
  - Communication channel is in the clear → a clever or well-positioned (in the network) attacker can read and potentially modify the information exchanged over the channel
    - Confidentiality problem that becomes an authentication problem

→ Encryption helps **mitigating** many of these problems

Suggested reading

- Bykova, Marina, and Shawn Ostermann. "Statistical analysis of malformed packets and their origins in the modern Internet." *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002.
- Hao Yang ; Osterweil, E. ; Massey, D. ; Songwu Lu ; Lixia Zhang. Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC. *IEEE Transactions on Dependable and Secure Computing. Vol 8, Issue 5.*
- Internet Census 2012. Port scanning /0 using insecure embedded devices.
  - http://internetcensus2012.bitbucket.org/paper.html
- Blackert, W. J., et al. "Analyzing interaction between distributed denial of service attacks and mitigation technologies." *DARPA information survivability conference and exposition, 2003. Proceedings*. Vol. 1. IEEE, 2003.
- S. M. Bellovin. 1989. Security problems in the TCP/IP protocol suite. *SIGCOMM Comput. Commun. Rev.* 19, 2 (April 1989), 32-48. DOI=http://dx.doi.org/10.1145/378444.378449

Useful network tools

- Wireshark / tcpdump → traffic monitoring
  - ARP requests
  - DNS requests
  - TCP 3-way handshake → SYN ACK
  - Network stack overview
- Nmap → scans (TCP; UDP; ..)
- Scapy → Python interface to generate network packets at the stack level
  - Manually craft network packets
- Other tools:
  - Ettercap → MitM attacks (ARP poisoning etc.)
  - Netcat → legacy tool to generate UDP/TCP traffic