Protocol Security

Giovanni Vigna
UCSB
Fall 2017

Protocol Security Analysis

- Modeling protocols
  - Representation (e.g., state-transition models)
  - Protocol runs
  - Attacker model
- Analyzing protocols
  - What are the assumptions?
  - Who is considered “trusted”?
    - Client side
    - Server side
  - What information can be sniffed/faked/forged/spoofed/hijacked?
- Exploiting protocol weaknesses
  - Stretching the limits
  - Violating the assumptions

Security Analysis of Protocol Implementations

- Protocol implementations that are inconsistent with respect to the design
- Protocol implementations that are buggy
- Protocol implementations that are misconfigured

Application Protocols

- Terminal/remote connection protocols (ssh)
- File transfer protocols (FTP, TFTP, scp, sftp)
- Mail protocols (SMTP, POP, IMAP)
- Web protocols (HTTP 1.0, 1.1, 2.0)
- Graphic protocols (X11, VNC/RFB)
- Domain Name System (DNS)
- Network management protocols (SNMP, BOOTP, DHCP, NTP)
- Routing protocols (RIP, OSPF, BGP)
- Session-level protocols (RPC, SSL)

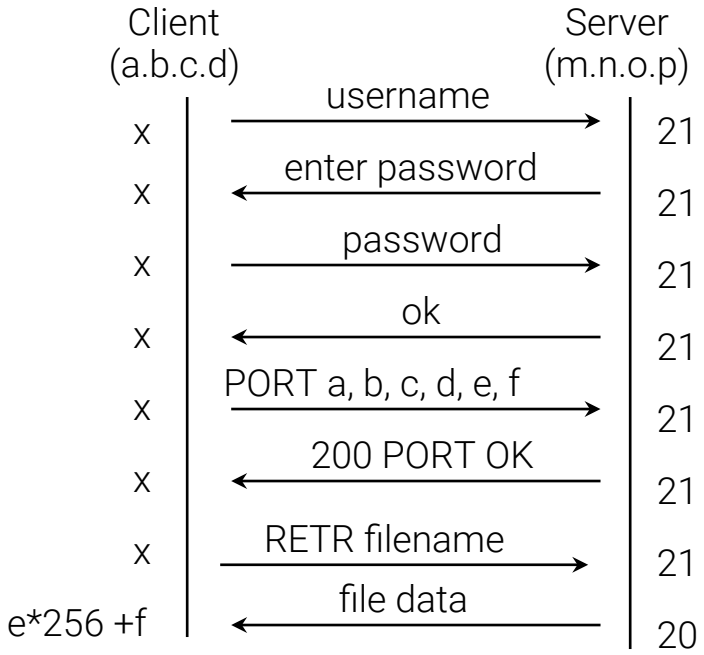
File Transfer Protocol

- FTP provides a file transfer service
- Based on TCP (RFC 172)
- The client (ftp) sends a connection request to the server (ftpd), listening on port 21
- The user provides username and password to authenticate himself/herself
- The user uses the GET and PUT commands to transfer files

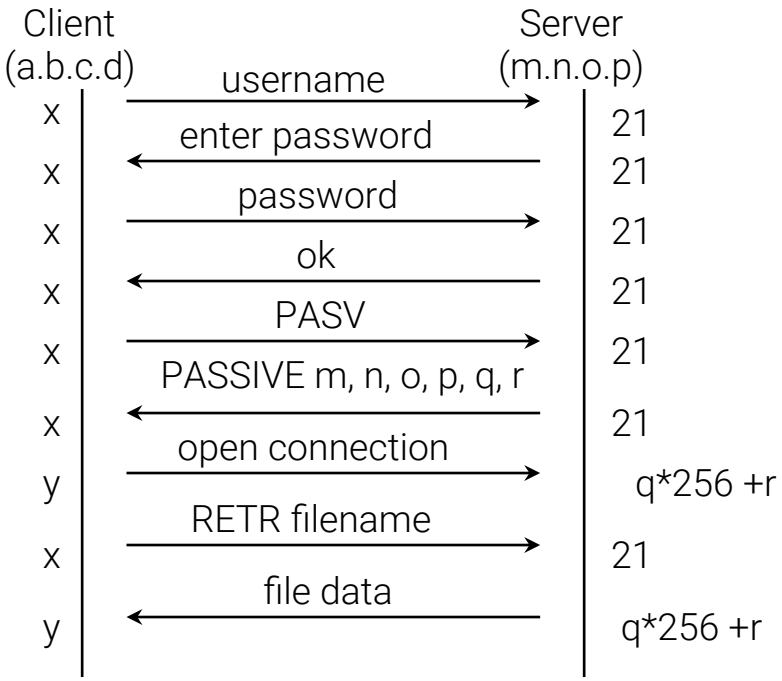
File Transfer Protocol

- Data is transferred on a separate TCP connection
  - The protocol specifies that one connection should be used for all the transfers
  - Most implementations use a different TCP connection for each transfer
- The client tells the server to connect to one of its local ports using the PORT command
- The server opens a connection from port 20 to the port specified by the client
- The file transfer is executed and the connection is closed

FTP Protocol



FTP Protocol



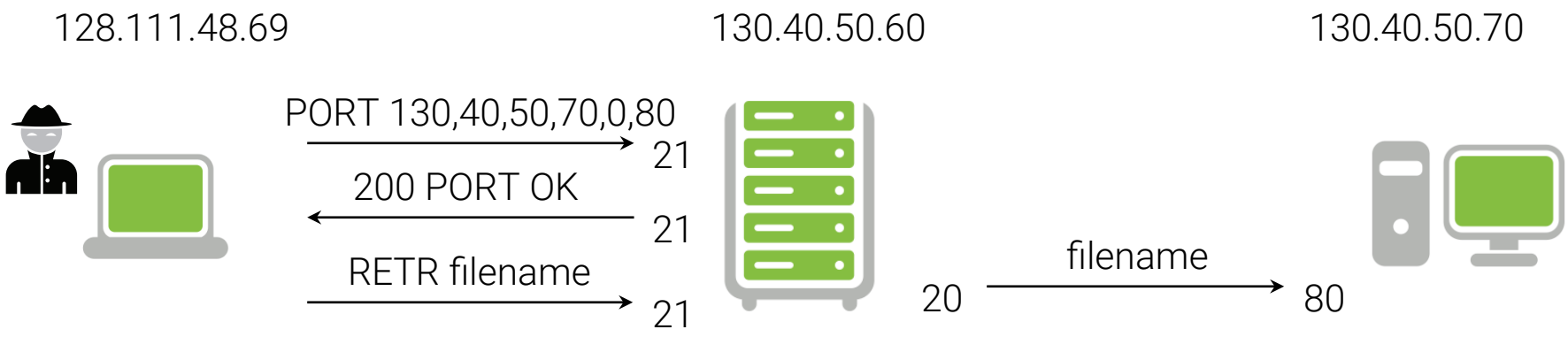
FTP Weaknesses

- The username and the password provided by the client is sent in the clear across the network
- All the session traffic is transferred without any protection
- Vulnerable to sniffing
- Vulnerable to hijacking
- Bounce attacks
- Anonymous FTP
  - Used to provide access to software/information
  - May be exploited/abused
    - Erroneous configuration (writable home, readable password file)
    - Bugs in the server implementation
    - Storage of pirated software, copyrighted data

FTP Bounce Attack

- The PORT command is used by the client to tell the server the address and port to be used when opening a data connection
- From RFC 959:
  It should be noted that the data port need not be in the same host that initiates the FTP commands via the control connection, but the user or the user-FTP process must ensure a “listen” on the specified data port.
- Therefore it is possible to instruct a server to open a connection to a third host

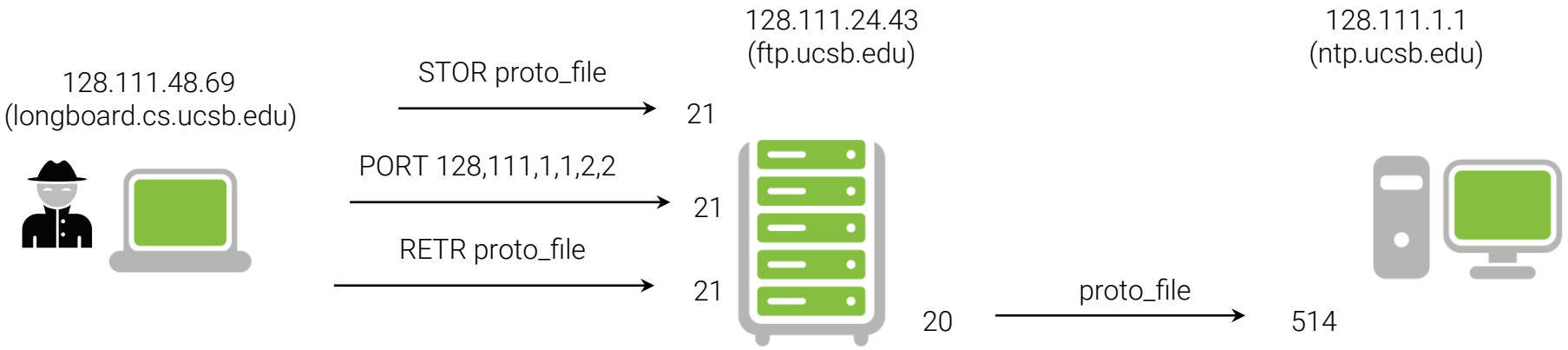
FTP Bounce Attack



FTP Bounce Attack

- Can be used to execute a TCP portscan
  - The host that appears to be the source of the scan is the FTP server
  - It is possible to scan a host that is behind the firewall protecting the FTP server
- Can be used to send data to arbitrary ports
  - If an FTP server has a writable directory, it is possible to upload a file and then have it delivered to a third host (at a port determined by the attacker) possibly bypassing the firewall or exploiting trust relationships
  - The same concept can be extended to bypass restrictions based on the IP address

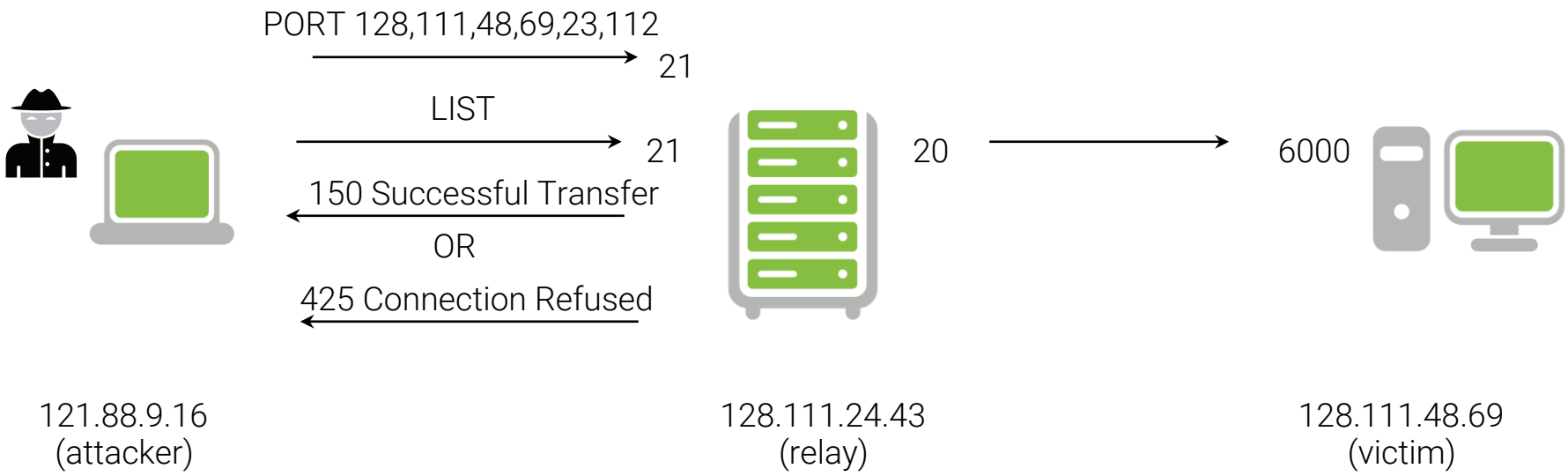
FTP Bounce Attack



proto_file

```
<null>root
<null>root
<null>echo toor:jD.E255EWCpoQ:0:0:toor:/:/bin/sh >> /etc/passwd
```

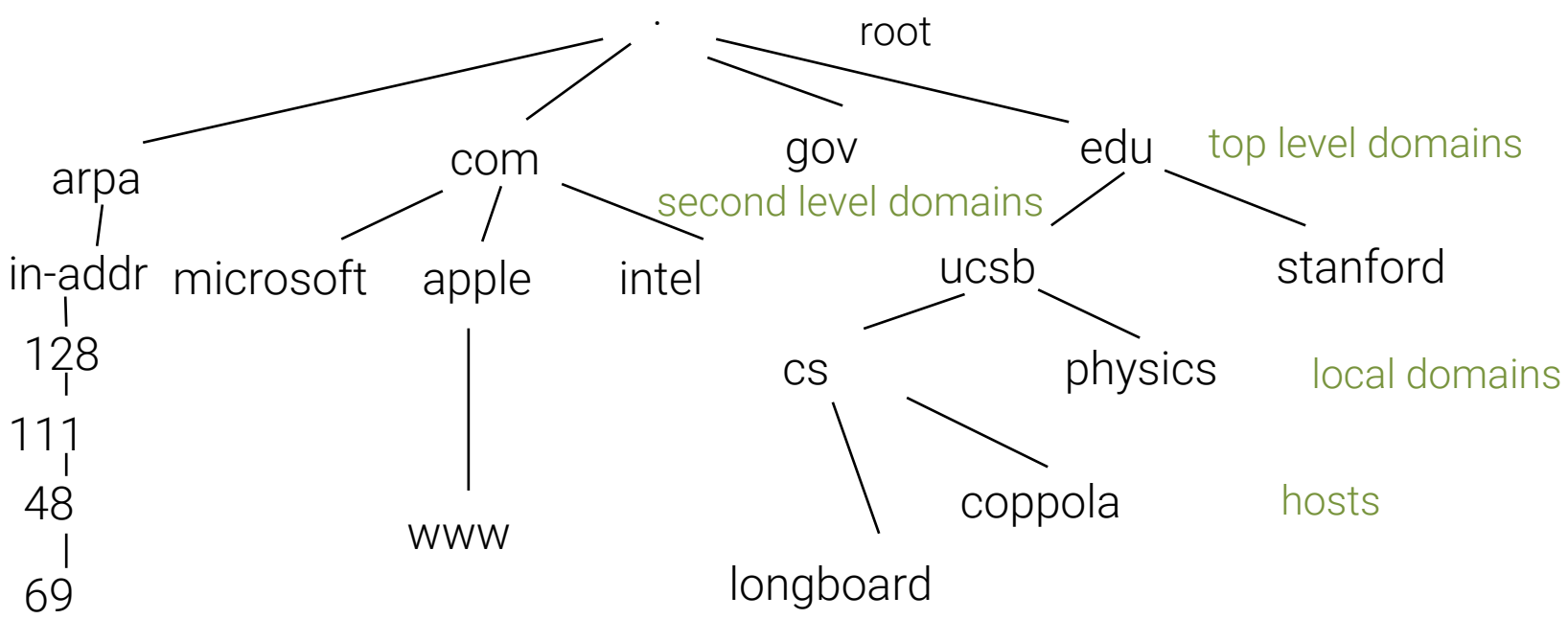
FTP Bounce Scan



The Domain Name System

- The DNS is a distributed database whose task is the mapping of IP addresses (128.111.48.69) into the corresponding names (longboard.cs.ucsb.edu) and vice versa
- The name space is hierarchically divided in domains
- Each domain is managed by a name server
- Clients access name server resolution services through the resolver library (gethostbyname(), gethostbyaddr())
- DNS uses mostly UDP and sometimes TCP for long queries and zone transfers between servers (port 53)

Domain Hierarchy



Address Mapping

- Mapping from IP addresses to names (pointer queries) uses a dedicated branch starting with ARPA.IN-ADDR
- An IP address of 128.111.48.69 is mapped into the name 69.48.111.128.in-addr.arpa
- Note that:
  - A name can have many IP addresses (e.g., to achieve load balancing)
  - An IP address can be associated with multiple names (e.g., in the case of a server hosting multiple web sites)

Name Servers

- Servers are responsible for mapping names in a zone
- Root servers are associated with the top of the hierarchy and dispatch queries to the appropriate domains
- Server types:
  - Primary: authoritative for the domain
  - Secondary: backup server, gets its data through zone transfers
  - Caching-only: relies on other servers to resolve names but maintains the results in a cache
  - Forwarding: simply forwards queries to other servers
  - Recursive: provides full resolution services (might be open)
- A server that cannot answer a query directly forwards the query up in the hierarchy
- The results are maintained in a local cache for a limited time (which can range from minutes to days)

DNS Clients

- DNS clients use the /etc/resolv.conf file (on UNIX) to find out what the servers for the domain are:

  domain cs.ucsb.edu
  nameserver 128.111.41.10

- On Linux, the DNS can be queried with a number of tools
  - host
  - dig

DNS Queries

- Recursive queries: require a name server to find the answer to the query itself
- Iterative queries: require a name server to provide the information requested or a reference to another server that may have the information
  - Note that in this case the reference to another server contains BOTH the name of the name server (e.g., ns.example.com) AND its IP address, to avoid circular dependencies (this is called “glue data”)

The Root Domain and Top-Level Domains

- The root domain is an implicit domain that contains all other domains (FQDNs should end with a period)
- There used to be a few TLDs (com, org, net, info, biz, etc. + country domains) but recently the ICANN allowed for 1000+ new TLDs
- In general, when queries cannot be answered directly, a root name server is involved (www.root-servers.org)
  - A: 198.41.0.4
  - B: 192.228.79.201
  - ...
- There are currently 13 root name servers
- If caching is done correctly, this type of requests should be infrequent

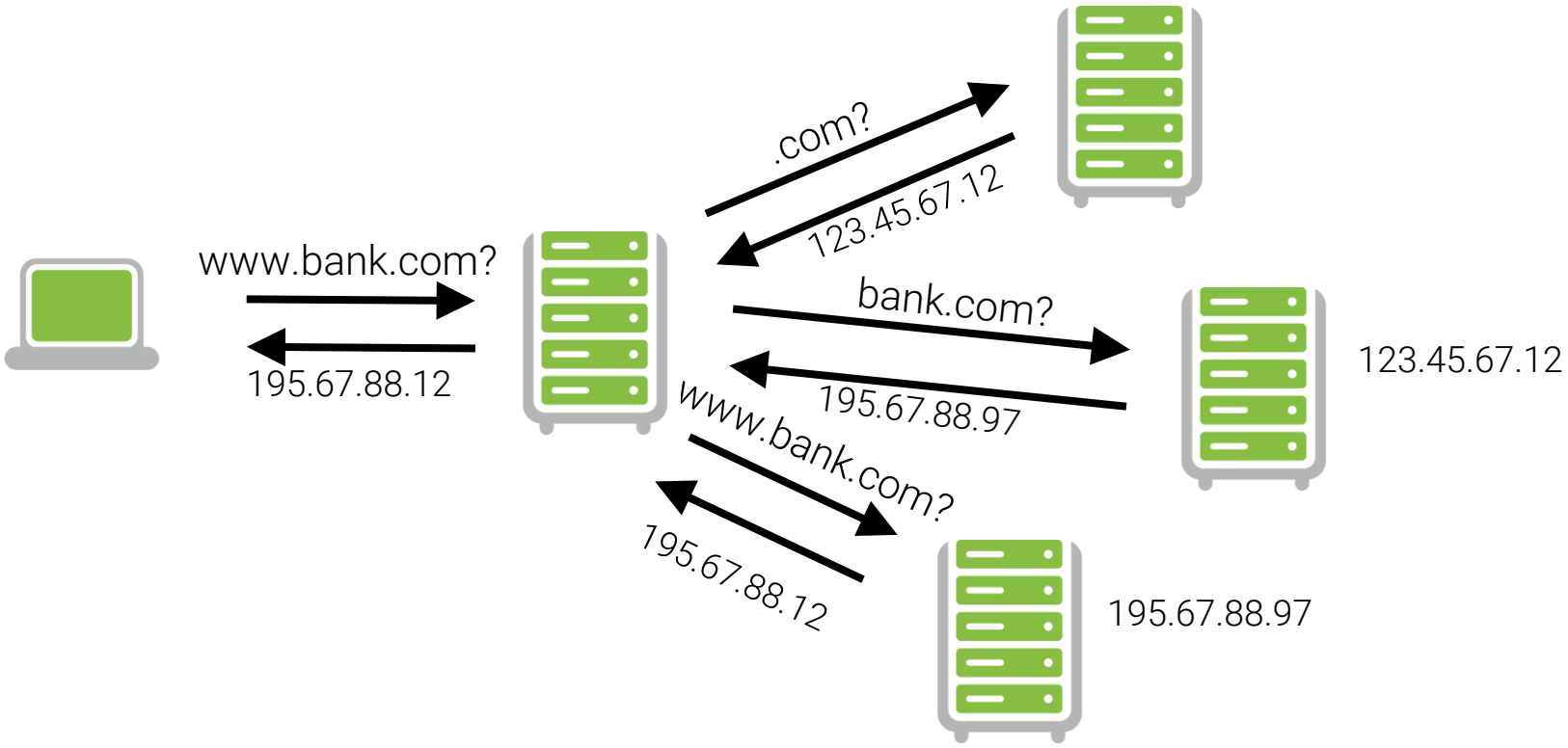
Root Name Servers

- The root name servers are a critical part of the infrastructure
- They are protected by anti-DOS systems and they use a number of high-availability techniques to prevent malfunctioning
- Incidents:
  - In October 21, 2002 there was the first substantial DOS attack against the root name servers (lasted an hour)
  - In February 6, 2007 a DOS attack lasted 24 hours
  - In February 12, 2012 an attack was threatened (possibly by Anonymous) but never performed

DNS Query

```
192.168.1.100.1044 > 4.2.2.1.53:  54061+ A? morgan.ece.ucsb.edu.
4.2.2.1.53 > 192.168.1.100.1044:  54061 1/4/4 A 128.111.21.1
192.168.1.100.1044 > 4.2.2.1.53:  54062+ PTR? 1.21.111.128.in-addr.arpa.
4.2.2.1.53 > 192.168.1.100.1044:  54062 1/5/5 PTR morgan.ece.ucsb.edu.
```

Anatomy of a DNS Request



Data

- The DNS data is structured in Resource Records
- Each RR contains
  - Name: domain this RR refers to
  - Type: type of record (e.g., A, MX, etc)
  - Class: “ IN” for IP
  - Time to live: specifies how long the RR will be kept in the cache (in seconds)
  - Data length: length of the data
  - Data contents: record-specific data

Record Types

- SOA Start Of Authority: specifies a domain
- NS Name Server: authoritative name server for a domain
- A Address: address for a host
- CNAME Canonical Name: real name for a host
- HINFO Host Information: CPU and OS type
- WKS Well Known Services: available services on a host

Record Types

- PTR Domain Name Pointer: reference to other names in the domain (used to map IP address to fully qualified domain names)
- MX Mail Exchanger: specifies mailer hosts (relays, etc)
- MB Mailbox: specifies the host used by a user to receive email
- MR Mail Rename: specifies alias for email
- MINFO Mailbox Information: specifies a mailing list
- MG Mail Group Member: specifies the users of a mailing list

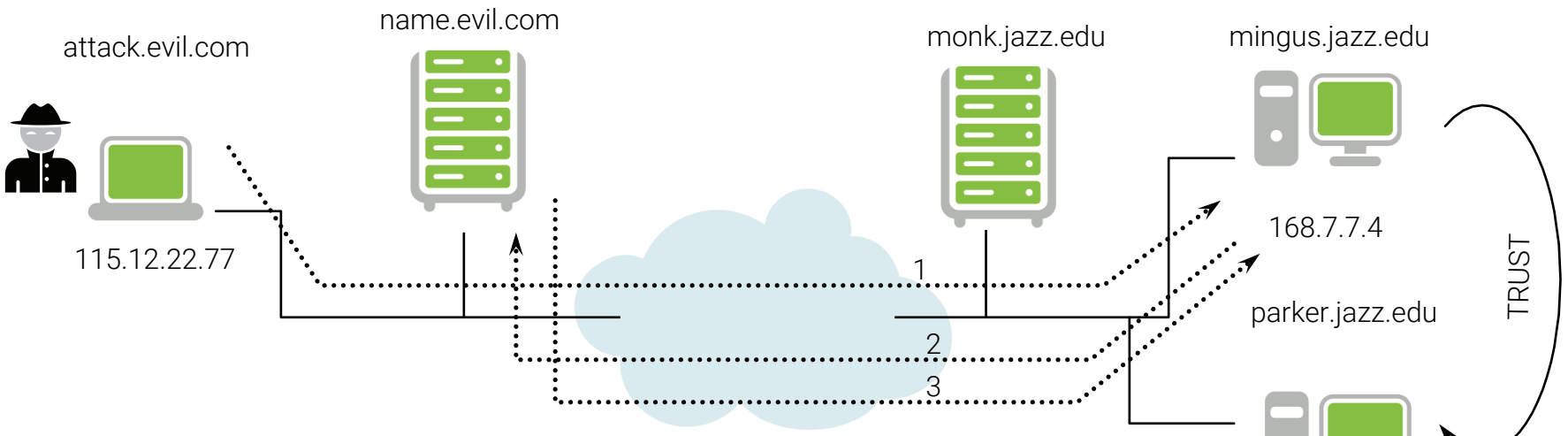
DNS Weaknesses

- The DNS may disclose too much information
  - (HINFO records, exhaustive queries, zone transfers)
- Being based on UDP, it is vulnerable to
  - Spoofing
  - Hijacking
- Too trusting implementations are vulnerable to cache-poisoning attacks
- Clients can be compromised and directed to malicious DNS servers
- Long replies can be used in denial-of-service attacks

DNS Zone Transfers

- DNS zone transfers are used by a client server to obtain a copy of the DNS zone records from a master server
- The client performs a SOA query to determine the current version of the zone data
- If the version of the client is not in sync with the current version, the client performs an AXFR (complete) or IXFR (incremental) query to update its zone
- The result is a large amount of information that can be used to prepare an attack

DNS Spoofing



1) Connection request from 115.12.22.77
2) Request for name associated with 115.12.22.77
3) Server answers 115.12.22.77 = parker.jazz.edu
4) Access is granted because of the trust relationship

Double Reverse Lookup

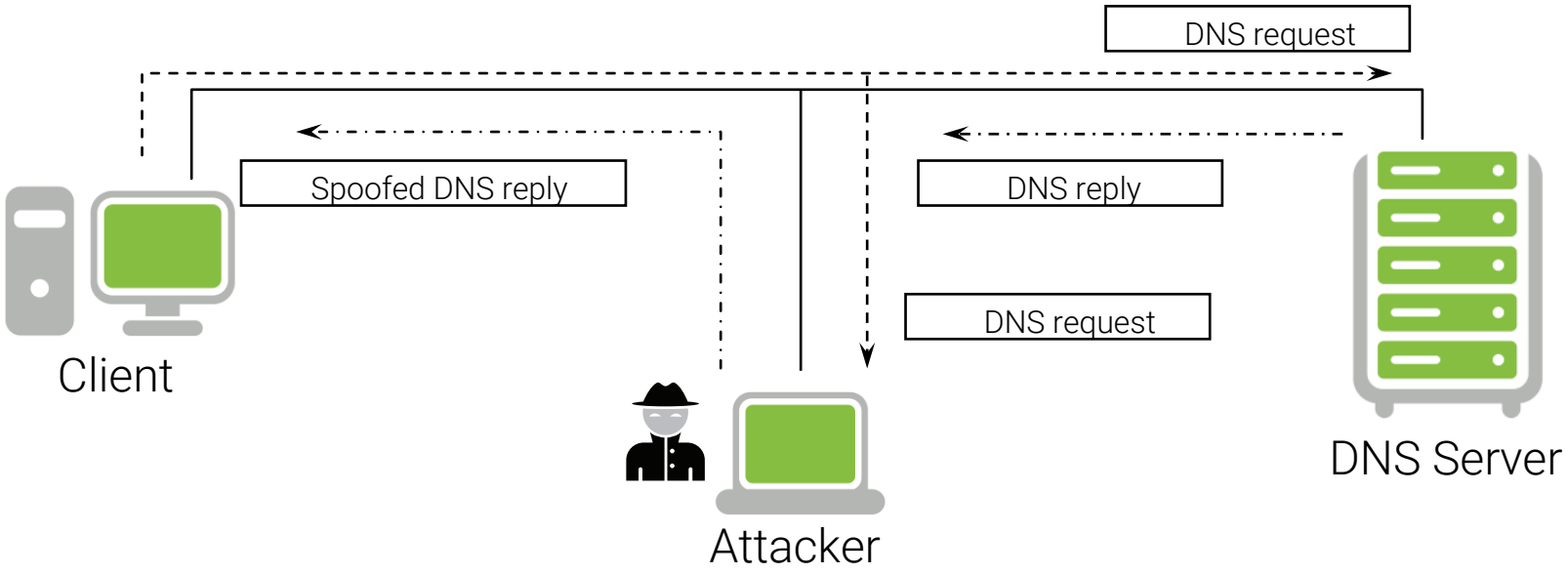
- Used to protect from DNS spoofing attacks
- Given the IP address addr1 obtain name n
- By using n obtain IP address addr2
- Check if addr1 = addr2

DNS Hijacking

- It is possible to perform DNS hijacking by
  - Racing with the server with respect to a client
  - Racing with a server with respect to another server
- In some cases, “blind” DNS hijacking requires to guess the request id
- Many implementations use sequential numbers

```
192.168.1.100.1044 > 4.2.2.1.53:  54061+ A? morgan.ece.ucsb.edu.
4.2.2.1.53 > 192.168.1.100.1044:  54061 1/4/4 A 128.111.21.1
192.168.1.100.1044 > 4.2.2.1.53:  54062+ PTR? 1.21.111.128.in-addr.arpa.
4.2.2.1.53 > 192.168.1.100.1044:  54062 1/5/5 PTR morgan.ece.ucsb.edu.
```

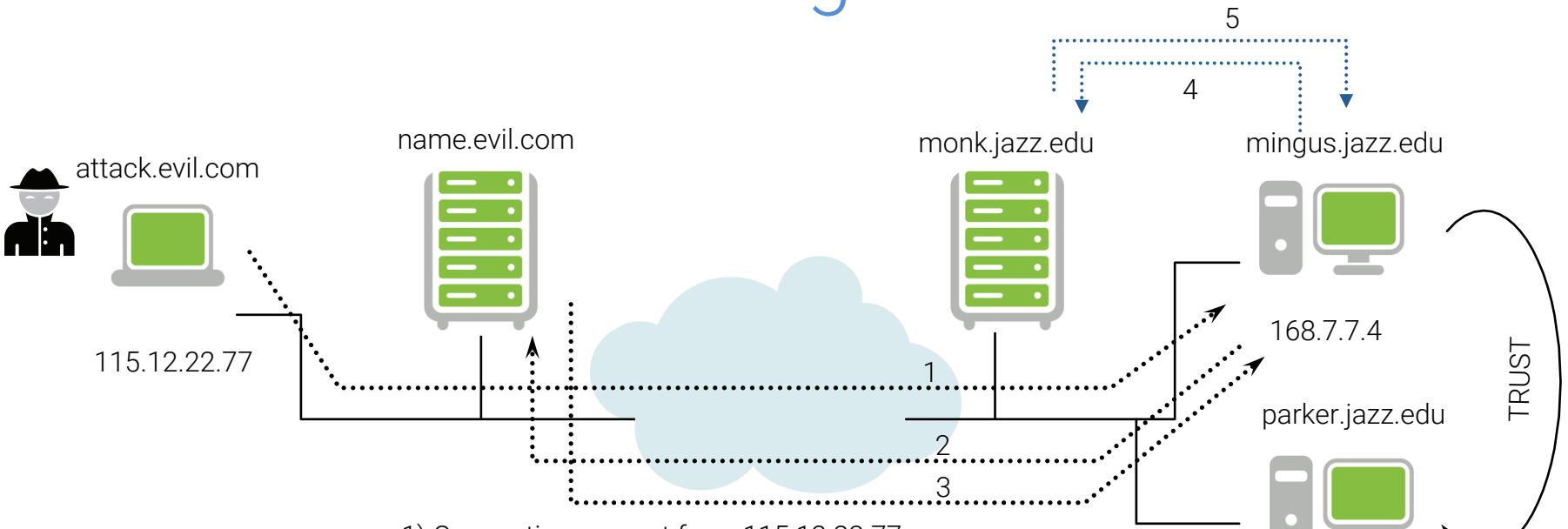
DNS Hijacking



DNS Cache Poisoning

- This attack exploits a bug in some DNS implementations
- A server stores in the cache anything that is contained in a DNS reply
- A malicious DNS server returns additional answers that will poison the cache
- Some old implementations will even accept answer records in DNS request, caching the information
- DNS cache poisoning can:
  - add a record for a specific host
  - add a record for the authoritative server for a whole domain

DNS Cache Poisoning



1) Connection request from 115.12.22.77
2) Request for name associated with 115.12.22.77
3) Server answers 115.12.22.77 = parker.jazz.edu
   and, in addition, parker.jazz.edu=115.12.22.77
4) mingus executes a reverse lookup for the IP of parker.jazz.edu
5) monk answers with the cached record
6) Access is granted because of the trust relationship

DNS Cache Poisoning Countermeasures

- Make sure that the additional information is relative to the domain being queried (bailiwick checking)
  - It is common to return, with an authoritative answer, the names of the DNS server(s) for the domain
- An attacker might try to answer queries for www.bank.com with a fake answer that provide additional information about the authoritative name servers for bank.com, which actually point to a malicious DNS server
- The malicious servers can then resolve future request for the address of any host in the bank.com zone
- However, the answer:
  - Must come from the right port
  - Must contain the right sequence number
- If the attacker loses the race, he will have to wait for the TTL to expire

The Kaminsky Attack

- Attack to poison the cache of a recursive DNS server
- Remote DNS cache poisoning through hijacking requires the attacker to guess the 16-bit ID value used to match requests to replies and the source port used in the request
  - ID used to be sequential and it is now random
  - Source port is most of the time not random
    - Randomness might be removed by NAT boxes

The Kaminsky Attack

- The idea is to create a “birthday attack” against the DNS
  - The attacker sends to the DNS server of a victim (e.g., ns.victim.com) multiple requests for hosts in the same domain of the target
    - The target domain is one that the attacker wants to control, e.g., bank.com
  - At the same time, the attacker sends a number of spoofed replies that appear to come from the authoritative DNS server of the target domain
    - Different from a classic spoofing attack that would send n spoofed replies for a single request

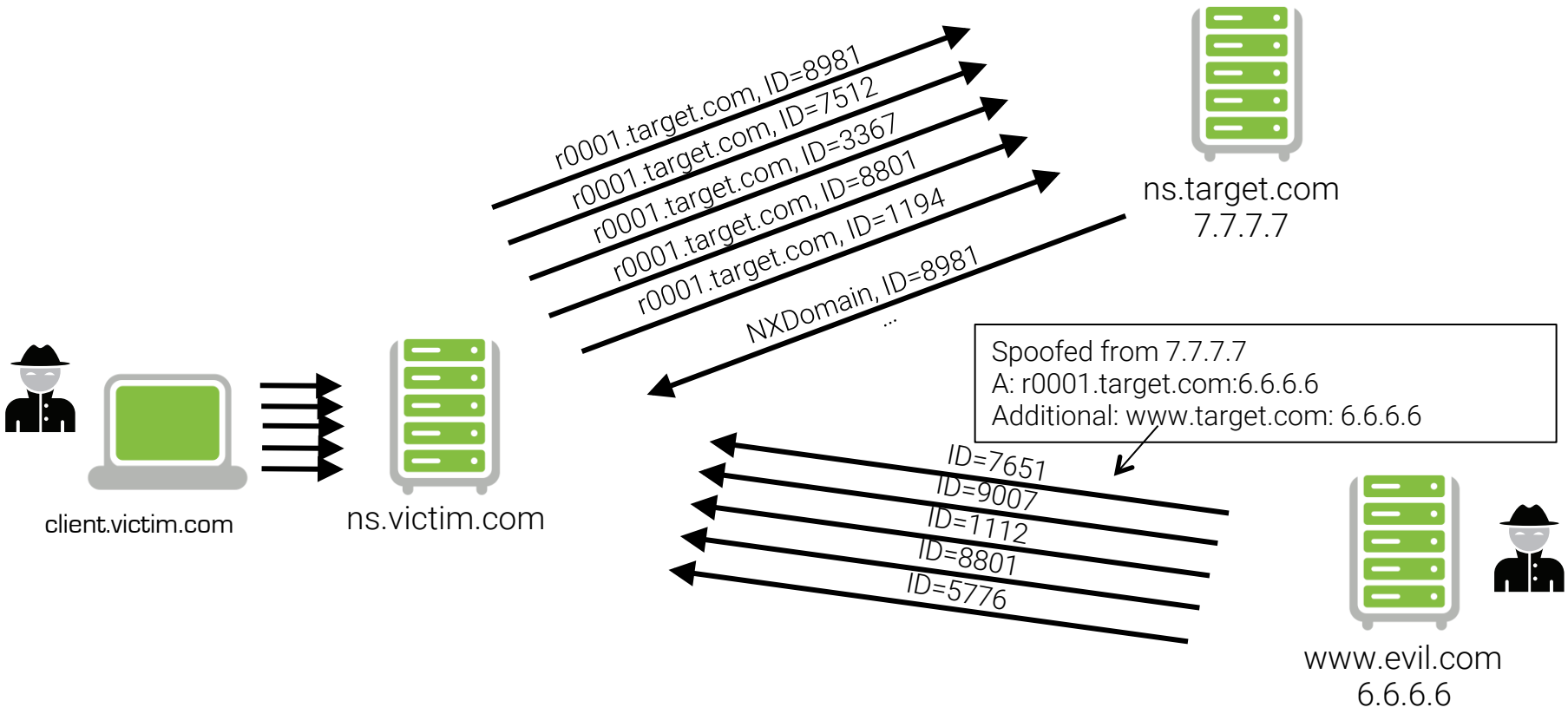
The Birthday Attack

- Given a function f, the goal of the attacker is to find two inputs x1 and x2 such as f(x1) = f(x2), called a collision
- If a function f yields results from a large set H with equal probability, after evaluating 1.25*sqrt(H) different arguments one has the 50% possibility of finding a collision
- Example: after how many tries I have a 50% probability of finding two people with the same birthday: 1.25*sqrt(365) = 23.88
  - In a room with 24 people there is a 50% probability that two persons have the same birthday
  - If H = 65536, 1.25 * sqrt(H) = 320

The Kaminsky Attack

- The attacker asks for random00001.bank.com multiple times to the victim DNS
- The victim DNS will generate a number of requests for the address of random00001.bank.com, each with a different query ID
- The attacker sends a set (~200) of spoofed replies with random query IDs and an additional resource record for www.bank.com that points to the attacker's host
  - Legitimate DNS will respond with "Non-existent domain"
- After a number of attempts, thanks to the birthday attack paradox, the attacker will finally win the race
- Since the piggybacked information is added in association with a request for a host in the same domain, it is accepted (it passes the bailiwick check)
- If the additional record is for the authoritative DNS server of bank.com, from this point on the attacker is able to control how any bank.com name is resolved by the victim

Birthday Attack Against DNS



DNSSEC

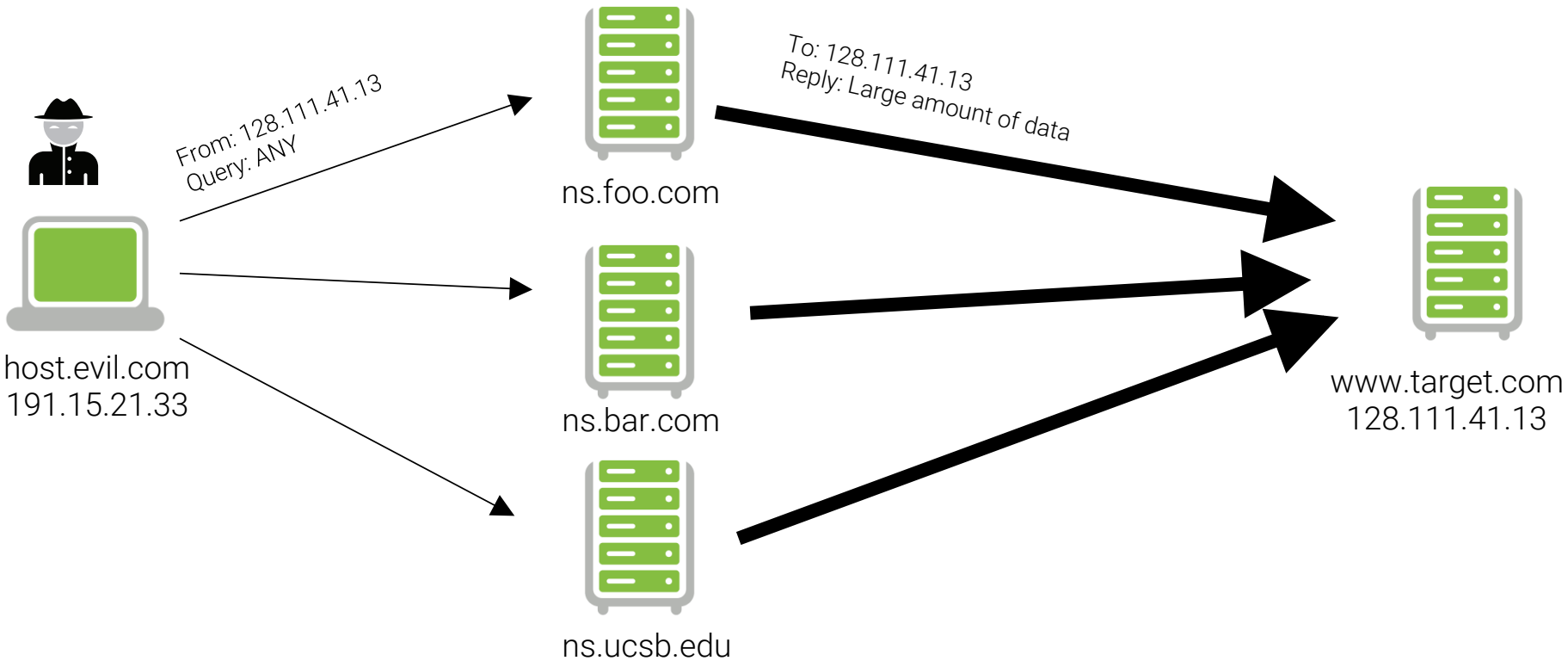
- DNSSEC: DNS Security Extensions
  - Protects against data spoofing and corruption
  - Authenticates servers, who sign records with their private keys

- DNSSEC requires a re-deployment of the Internet infrastructure
- DNSSEC introduces substantial overhead

DNS Amplification

- Open resolvers are DNS servers that will respond to queries from any IP address
- ANY queries ask all information about a domain
- An attacker uses spoofed small ANY queries against open resolvers to generate large replies
  - Situation made worse by DNSSEC data, which is large

DNS Amplification



DNS Tunneling

- In many situation in which traffic is otherwise restricted, DNS queries are allowed
  - For example, in pay-per-use WiFi
- If one can control a DNS server it is possible to established a tunnel that leverages DNS
- Information can be included in the name of a host (e.g., encoded in base32 – non case-sensitive)
- Information can be included in TXT records in the server response

Secure Socket Layer (SSL)

- SSL is a session level protocol that provides a generic security communication channel
- Placed between TCP and the application protocol
- Developed by Netscape
- A history of security issues
- Latest version is 3.0, which has been deprecated in 2015

Transport Layer Security (TLS)

- SSL was standardized by the IETF and renamed to Transport Layer Security (TLS)
  - Current version is 1.2
  - Version 1.3 is in draft format, as of October 2017

Properties

- Encrypted communication with symmetric encryption
- Use of Message Authentication Codes to guarantee integrity
- Two-way authentication using certificates

TLS Session

- TLS is stateful, that is, client and server maintain a number of configuration parameters that can be reused for many connections
  - Identifiers
  - X.509 certificates of the communication partners
  - Compression algorithms
  - Cipher spec, specifying the algorithms to be used to establish keys, encrypt data, and generate message authentication codes (MACs)
  - Master secret, a secret shared between client and server

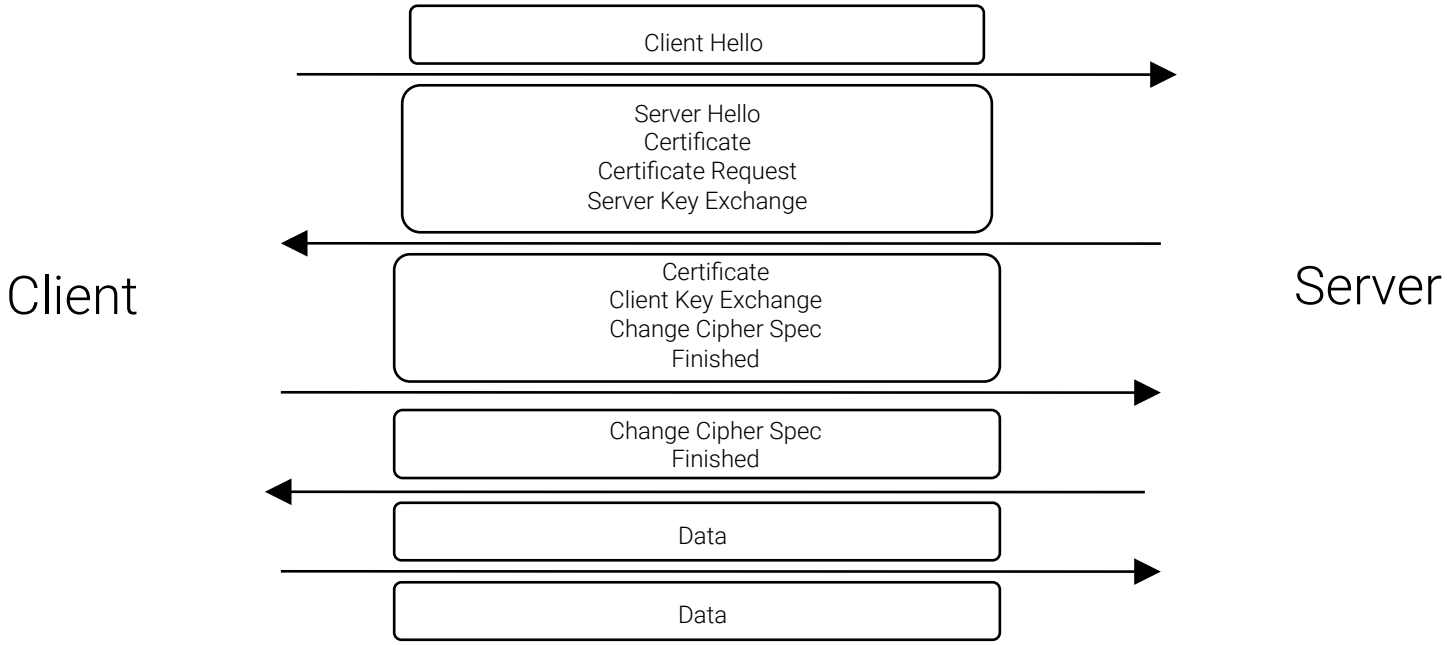
TLS Connection

- Each TLS connection is characterized by:
  - Two random numbers chosen by client and server
  - Keys used to generate MACs
  - Keys used to encrypt traffic
  - Sequence numbers for sent and received messages

TLS Stack

- Two levels
  - TLS Handshake Protocol: manages the establishment of the cipher spec of a session
  - TLS Record Protocol: processes data coming from applications by applying encryption and authentication procedures

Handshake Protocol



Handshake Protocol: ClientHello

- Used to determine the cipher suite
- Initially the cipher suite is TLS_NULL_WITH_NULL_NULL
- The client sends a client hello message to the server, containing:
  - A random number
  - A session identifier that, if non null, refers to a previous session to be reused
  - A list of cipher suites supported by the client (e.g., TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256)
  - A list of compression methods supported by the client

Handshake Protocol: ServerHello

- The server replies with a server hello message containing:
  - A random number
  - A session identifier (if it is equal to the one proposed by the client then the referenced session will be reused, otherwise the identifier specifies a new session)
  - A cipher suite among those proposed by the client
  - A compression algorithm among those proposed by the client

Handshake Protocol: Key Exchange

- Once the client and the server have agreed upon a cipher suite they exchange their respective certificates
  - Server uses the Certificate message
  - If client-side authentication is required, the server sends a CertificateRequest message
- Using the associated public keys client and server agree on a common encryption key
  - The client uses a ClientKeyExchange to send a PreMasterSecret key or a public key
- The initial random numbers and the PreMasterSecret are used to generate a MasterSecret, which is then used to generate all the other keys
- If everything goes ok they send each other a ChangeCipherSpec message and they switch to encrypted communication

Record Protocol

- Receives messages from applications
- Splits messages in segments of max 16 Kbyte
- Compresses the segments using the algorithm specified in the session descriptor
- Generates a MAC for the message
- Encrypts the message using the algorithm specified in the session descriptor
- Sends the message

TLS/SSL Tools

- The openssl tool suite allows one to perform a number of SSL/TLS-related operations from the command line
  - Generation of keys and certificates
  - Testing of TLS/SSL connections
  - Encryption/decryption of messages
  - S/MIME message handling
- The stunnel tool (http://stunnel.org) can be used to add TLS/SSL support to clients and servers without having to modify their code
- SSLdump (http://www.rtfm.com/ssldump/) parses TLS/SSL sessions and can decrypt the records exchanged in an TLS/SSL connection (if the right key material is provided)

TLS/SSL Attacks

- BEAST Attack (2011)
  - Uses a chosen-plaintext attack to break the crypto
  - Requires a malicious client-side component that makes requests
  - http://vnhacker.blogspot.com/2011/09/beast.html
- POODLE Attack (2014)
  - Padding Oracle On Downgraded Legacy Encryption
  - MITM attack that force the parties to downgrade from TLS to SSL 3.0 and decrypt traffic

TLS/SSL Attacks

- Heartbleed (2014)
  - Bug in the OpenSSL implementation
  - Missing size check on a field in a Heartbeat request discloses information stored in memory (including private keys of the server)
  - http://heartbleed.com/
  - https://xkcd.com/1354/
- FREAK Attack (2015)
  - Factoring Attack on RSA-EXPORT Keys
  - Some SSL implementation can be convinced to roll back to more insecure protocols
    - Introduced in the 1990s to adhere to export controls on crypto

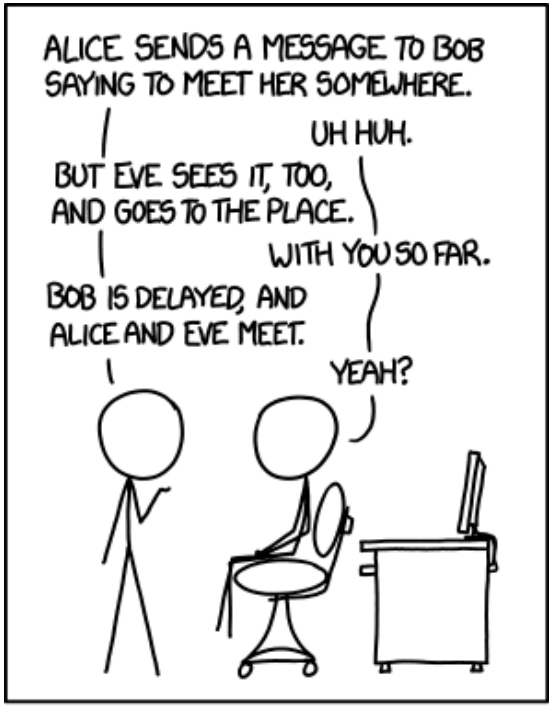
SSL/TLS Attacks

- RC4 Biases (2015)
  - The paper "On the Security of RC4 in TLS and WPA", published in the USENIX Security Symposium in July 2013 highlighted problems in the use of RC4
  - By exploiting biases in the first 256 bytes produced by the RC4 algorithm it is possible to extract plaintext from cypher-text
  - In February 2015 the IETF published RFC 7465 prohibiting the use of RC4 in TLS

Conclusions

- Designing protocols that are secure and perform well is hard
- Many old protocols are still used in many situations
- Attacks include
  - Sniffing
  - Spoofing
  - Hijacking
  - Amplification
  - Abuse to bypass protection
  - Decryption of encrypted traffic

Questions?



http://xkcd.com/1323/