CS 88: Security and Privacy

19: DNS and UDP

04-11-2024

slides adapted from Dave Levine, Jim Kurose
Reading Quiz
A “Simple” Task

Send information from one computer to another
Not really so simple…
We only need...

- Manage complexity and scale up
- Naming and addressing
- Moving data to the destination
- Reliability and fault tolerance
- Resource allocation, Security, Privacy..
Five-Layer Internet Model

- **Application**: the application (e.g., the Web, Email)
- **Transport**: end-to-end connections, reliability
- **Network**: routing
- **Link (data-link)**: framing, error detection
- **Physical**: 1’s and 0’s/bits across a medium (copper, the air, fiber)
OSI 5 Layer Model

**Network**

Responsible for packet forwarding. How to get a packet to the final destination when there are many hops along the way.

**Data Link**

How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.

**Physical**

How do bits get translated into electrical, optical, or radio signals?
OSI 5 Layer Model

Transport

Allows a client to establish a connection to specific services (e.g., web server on port 80). Provides reliable communication.

Network

Responsible for packet forwarding. How to get a packet to the final destination when there are many hops along the way.

Data Link

How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.

Physical

How do bits get translated into electrical, optical, or radio signals
OSI 5 Layer Model

**Application**

Defines how individual applications communicate. For example, **HTTP** defines how browsers send requests to web servers.

**Transport**

Allows a client to establish a connection to specific services (e.g., web server on port 80). Provides reliable communication.

**Network**

Responsible for packet forwarding. How to get a packet to the final destination when there are many hops along the way.

**Data Link**

How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.

**Physical**

How do bits get translated into electrical, optical, or radio signals?
OSI 5 Layer Model

**Network**
- Responsible for packet forwarding. How to get a packet to the final destination when there are many hops along the way.

**Data Link**
- How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.

**Physical**
- How do bits get translated into electrical, optical, or radio signals
## OSI 5 Layer Model

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transport</strong></td>
<td>Allows a client to establish a connection to specific services (e.g., web server on port 80). Provides reliable communication.</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Responsible for packet forwarding. How to get a packet to the final destination when there are many hops along the way.</td>
</tr>
<tr>
<td><strong>Data Link</strong></td>
<td>How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.</td>
</tr>
<tr>
<td><strong>Physical</strong></td>
<td>How do bits get translated into electrical, optical, or radio signals</td>
</tr>
</tbody>
</table>
OSI 5 Layer Model

- **Application**: Defines how individual applications communicate. For example, **HTTP** defines how browsers send requests to web servers.

- **Transport**: Allows a client to establish a connection to specific services (e.g., web server on port 80). Provides reliable communication.

- **Network**: Responsible for packet forwarding. How to get a packet to the final destination when there are many hops along the way.

- **Data Link**: How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.

- **Physical**: How do bits get translated into electrical, optical, or radio signals.
Layering and encapsulation

Layer
- Application
- Transport: reliability
- Network: routing
- Link: framing, error detection

Physical

Layer Diagram:
- Ethernet
- IP
- TCP/UDP
- Data
- Port no.
- IP address
- MAC address
- Application
- Transport: reliability
- Network: routing
- Link: framing, error detection
Abstraction!

• Hides the complex details of a process

• Use abstract representation of relevant properties make reasoning simpler

• Ex: Alice and Bob’s knowledge of postal system:
  • Letters with addresses go in, come out other side
TCP/IP Protocol Stack

- Application Layer
  - HTTP

- Transport Layer
  - TCP

- Network Layer
  - IP
  - router

- Link Layer
  - Ethernet interface
  - SONET interface
  - host
  - router
TCP/IP Protocol Stack
The “End-to-End” Argument

Don’t provide a function at lower layer if you have to do it at higher layer anyway ... 
... unless there is a very good performance reason to do so.

Examples: error control, quality of service

Basic security goals:

- **Confidentiality**: No one should be able to read our data/communications unless we want them to.
- **Integrity**: No one can manipulate our data/communications unless we want them to.
- **Availability**: We can access our data/communication capabilities when we want to.
Network Attacks: Classes of Attackers

- **MiTM**: Can see packets, and can modify and drop packets
- **On-path**: Can see packets, but can’t modify or drop packets
- **Off-path**: Can’t see, modify, or drop packets

Which type of attacker is more powerful?
A. on-path
B. off-path
C. neither is strictly stronger than the other
DNS: a distributed, hierarchical database

- com DNS servers
  - yahoo.com DNS servers
  - amazon.com DNS servers

- org DNS servers
  - pbs.org DNS servers

- edu DNS servers
  - swarthmore.edu DNS servers
  - umass.edu DNS servers
    - cs.swarthmore.edu DNS servers
      - allspice.cs.swarthmore.edu Host
      - (other cs hosts)

- allspice.cs.swarthmore.edu

Nameless root, Usually implied.
Goals of DNS

A wide-area distributed database
Possibly biggest such database in the world!

Goals
- Scalability; decentralized maintenance
- Robustness
- Global scope
- Names mean the same thing everywhere
- Distributed updates/queries
- Good performance
DNS: Application Layer Protocol

• distributed database
  • implemented in hierarchy of many name servers.

• application-layer protocol:
  • hosts communicate to name servers
  • resolve names → addresses

• **Core Internet function, implemented as application-layer protocol**
DNS: Domain Name System

**People:** many identifiers:
- name, swat ID, SSN, passport #

**Internet hosts (endpoints), routers (devices inside a n/w):**
- “name”, e.g., www.google.com - used by humans
- IP address (32 bit) - used for addressing packets

How do we map between IP address and name, and vice versa?
DNS: Root Name Servers

- Root name server:
  - Knows how to find top-level domains (.com, .edu, .gov, etc.)
  - How often does the location of a TLD change?
  - approx. 400 total root servers
  - Significant amount of traffic is not legitimate

13 root name “servers” worldwide
DNS: a distributed, hierarchical database

Root DNS Servers

- com DNS servers
  - yahoo.com DNS servers
  - amazon.com DNS servers
- org DNS servers
  - pbs.org DNS servers
- edu DNS servers
  - swarthmore.edu DNS servers
  - umass.edu DNS servers
    - cs.swarthmore.edu DNS servers
      - allspice.cs.swarthmore.edu Host
    (other cs hosts)
DNS: a distributed, hierarchical database

- Root DNS Servers
  - ... ...
  - com DNS servers
  - org DNS servers
  - edu DNS servers
- yahoo.com DNS servers
- amazon.com DNS servers
- pbs.org DNS servers
- swarthmore.edu DNS servers
- umass.edu DNS servers
- cs.swarthmore.edu DNS servers
- allspice.cs.swarthmore.edu Host

(other cs hosts)
Authoritative DNS servers:

- Organization’s own DNS server(s), providing authoritative hostname to IP mappings for organization’s named hosts
- Can be maintained by organization or service provider, easily changing entries
- Often, but not always, acts as organization’s local name server (for responding to look-ups)
Local DNS Name Server

• Each ISP (residential ISP, company, university) has (at least) one
  • also called “default name server”

• When host makes DNS query, query is sent to its local DNS server
  • has local cache of recent name-to-address translation pairs (but may be out of date!)
  • acts as proxy, forwards query into hierarchy
Uses of DNS

Hostname to IP address translation
- Reverse lookup: IP address to hostname translation

Host name aliasing: other DNS names for a host
- Alias hostnames point to canonical hostname

Email: look up domain’s mail server by domain name
Different DNS Mappings

1-1 mapping between domain name and IP addr
www.cs.cornell.edu maps to 132.236.207.20

Multiple domain names maps to the same IP addr
eecs.mit.edu and cs.mit.edu both map to 18.62.1.6

Single domain name maps to multiple IP addrs
aol.com and www.aol.com map to multiple IP addrs

Some valid domain names don’t map to any IP addr
cmcl.cs.cmu.edu
DNS name resolution example #1

• allspice wants IP address for gaia.cs.umass.edu

iterative query:
• contacted server replies with name of server to contact
• “I don’t know this name, but ask this server”
DNS Packet

- Lightweight exchange of *query* and *reply* messages, both with the same message format.
- Primarily uses UDP for its transport protocol, which is what we’ll assume.
- Frequently, both clients and servers use port 53.
DNS requests sent over UDP

Four sections: questions, answers, authority, additional records

Query ID:
16 bit random value
Links response to query
Request

```
src port = 53
dst port = 53
```

```
src IP = 68.94.156.1
dst IP = 192.26.92.30
```

```
QID = 43561
```

```
Question count = 1
```

```
OP=0 - standard query
QR=0 - this is a query
```

```
RD=1 - recursion desired
```

```
What is A record for www.unixwiz.net?
```
**Response**

```
src IP = 192.26.92.30
src port = 53
QID = 43561
Question count = 1
Authority count = 2
QR=1 - this is a response
AA=0 - not authoritative
RA=0 - recursion unavailable

Qu: What is A record for www.unixwiz.net?

Au: unixwiz.net NS = linux.unixwiz.net 2 dy
Au: unixwiz.net NS = cs.unixwiz.net 2 dy
Ad: linux.unixwiz.net A = 64.170.162.98 1 hr
Ad: cs.unixwiz.net A = 8.7.25.94 1 hr
```
Authoritative Response

![IPv4 packet format diagram]

- **src IP**: 64.170.162.98
- **dst IP**: 68.94.156.1
- **src port**: 53
- **dst port**: 5798

**QID**: 43562
- **QR=1**: this is a response
- **AA=1**: Authoritative!
- **RA=0**: recursion unavailable

**Query**: What is an A record for www.unixwiz.net?
- **An**: www.unixwiz.net A = 8.7.25.94 1 hr
- **An**: unixwiz.net NS = linux.unixwiz.net 2 dy
- **An**: unixwiz.net NS = cs.unixwiz.net 2 dy
- **Ad**: linux.unixwiz.net A = 64.170.162.98 1 hr
- **Ad**: cs.unixwiz.net A = 8.7.25.94 1 hr

**Hosts**:
- **linux.unixwiz.net**
- **dnsr1.sbcglobal.net**
Common Security Assumptions

• Attackers can interact with our systems without particular notice.

• *Probing* (poking at systems) may go unnoticed ...
  • even if highly repetitive, leading to crashes, and *easy to detect*
  • Attackers can obtain access to a copy of a given system to measure and/or determine how it works

• It’s easy for attackers to know general information about their targets:
  • OS types, software versions, usernames, server ports, IP addresses, usual patterns of activity, administrative procedures

(Note, these tend to be pessimistic ... but prudent)
Common Security Assumptions

• Attackers can make use of automation – they can often find clever ways to automate

• Attackers can pull off complicated coordination across a bunch of different elements/systems

• Attackers can bring large resources to bear if needed – computation, network capacity
  • But they are not super-powerful (e.g., control entire Internet Service Providers)
DNS security

DNS Vulnerabilities:
• No authentication
• Connectionless transport layer protocol (UDP)

DNS Attacks:
• Amplification Attack
• Cache Poisoning
• Man-in-the-middle
• DNS Redirection
• DDoS
• DNS Injection
Attacking DNS

**DDoS attacks**
- Bombard root servers with traffic
  - Not successful to date
  - Traffic Filtering
  - Local DNS servers cache IPs of TLD servers, bypassing root
- Bombard TLD servers
  - Potentially more dangerous

**Redirect attacks**
- Man-in-middle
  - Intercept queries
- DNS cache poisoning
  - Send bogus replies to DNS server that caches

**Exploit DNS for DDoS**
- Send queries with spoofed source address: target IP
- Requires amplification
DNSSEC Hierarchy of Trust

Where is bankofamerica.com?

Root Zone (ICANN)

.com (Verisign)

dns.evil.com
dns.bofa.com

IP: 66.66.66.93
Key: < >
SIG: 9na8x7040a3
Solution: DNSSEC

• Cryptographically sign critical resource records
  • Resolver can verify the cryptographic signature

• Two new resource types
  • Type = DNSKEY
    • Name = Zone domain name
    • Value = Public key for the zone
  • Type = RRSIG
    • Name = (type, name) tuple, i.e. the query itself
    • Value = Cryptographic signature of the query results

Prevents hijacking and spoofing
Creates a hierarchy of trust within each zone
How many answers
Time to live in seconds
How many additional records?

$ dig @a.root-servers.net www.freebsd.org +noreturse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57494
;; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
www.freebsd.org. IN A

;; AUTHORITY SECTION:
org. 172800 IN NS b0.org.afilias-nst.org.
org. 172800 IN NS d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
b0.org.afilias-nst.org. 172800 IN A 199.19.54.1
d0.org.afilias-nst.org. 172800 IN A 199.19.57.1
$ dig @a.root-servers.net www.freebsd.org +norecurse

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57494
;; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.freebsd.org. IN A

;; AUTHORITY SECTION:
;org. 172800 IN NS b0.org.afilias-nst.org.
;org. 172800 IN NS d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
b0.org.afilias-nst.org. 172800 IN A 199.19.54.1
d0.org.afilias-nst.org. 172800 IN A 199.19.57.1

How many answers
Time to live in seconds
How many additional records?
16-bit transaction identifier that enables the DNS client to match up the reply with its original request.
the question we asked the server
type of response A = IP address, and NS = name server

Glue records
$ dig @199.19.54.1 www.freebsd.org +norecurse

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39912
;; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
;www.freebsd.org. IN A

;; AUTHORITY SECTION:
freebsd.org. 86400 IN NS ns1.isc-sns.net.
freebsd.org. 86400 IN NS ns2.isc-sns.com.
freebsd.org. 86400 IN NS ns3.isc-sns.info.
How many answers?

What does it mean if the answer section is empty?

How many additional records?
$ dig @ns1.isc-sns.net www.freebsd.org +norecurse

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17037
;; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.freebsd.org. IN A

;; ANSWER SECTION:
www.freebsd.org. 3600 IN A 69.147.83.33

;; AUTHORITY SECTION:
freebsd.org. 3600 IN NS ns2.isc-sns.com.
freebsd.org. 3600 IN NS ns1.isc-sns.net.
freebsd.org. 3600 IN NS ns3.isc-sns.info.

;; ADDITIONAL SECTION:
ns1.isc-sns.net. 3600 IN A 72.52.71.1
ns2.isc-sns.com. 3600 IN A 38.103.2.1
ns3.isc-sns.info. 3600 IN A 63.243.194.1

(authoritative for freebsd.org.)

How many answers?
what does the answer tell us

How many authoritative records?
what does the authority tell us?

How many additional records?
$ dig @ns1.isc-sns.net www.freebsd.org +norecurse
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17037
;; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.freebsd.org. IN A

;; ANSWER SECTION:
www.freebsd.org. 3600 IN A 69.147.83.33

;; AUTHORITY SECTION:
freebsd.org. 3600 IN NS ns2.isc-sns.com.
freebsd.org. 3600 IN NS ns1.isc-sns.net.
freebsd.org. 3600 IN NS ns3.isc-sns.info.

;; ADDITIONAL SECTION:
ns1.isc-sns.net. 3600 IN A 72.52.71.1
ns2.isc-sns.com. 3600 IN A 38.103.2.1
ns3.isc-sns.info. 3600 IN A 63.243.194.1

(authoritative for freebsd.org.)

How many answers?
How many authoritative records?
How many additional records?
Protocol Layering

Networks use a stack of protocol layers
- Each layer has different responsibilities.
- Layers define abstraction boundaries

Lower layers provide services to layers above
- Don’t care what higher layers do

Higher layers use services of layers below
- Don’t worry about how the layer below works

Application Layer
Transport: end-to-end connections, reliability
Network: routing
Link (data-link): framing, error detection
Physical: 1’s and 0’s/bits across a medium (copper, the air, fiber)
Transport Layer perspective

Networks use a stack of protocol layers
- Each layer has different responsibilities.
- Layers define abstraction boundaries

Lower layers provide services to layers above
- Don’t care what higher layers do

Higher layers use services of layers below
- Don’t worry about how the layer below works

Application is the boss

Transport: executing within the OS kernel

Network: ours to command!
Layering and encapsulation

Layer
Application
Transport: reliability
Network: routing
Link: framing, error detection
Physical
Ports: An Analogy

- Alice is pen pals with Bob. Alice’s roommate, Carol, is also pen pals with Bob.
- Bob’s replies are addressed to the same global (IP) address.
  - How can we tell which letters are for Alice and which are for Bob?
- Solution: Add a room number (port number) inside the letter.
  - In private homes, usually a port number is meaningless.
  - But, in public offices (servers), like Cory Hall, the port numbers are constant and known.
Each application on a host is identified by a *port number*

TCP connection established between port A on host X to port B on host Y. Ports are 1–65535 (16 bits)

Some destination port numbers used for specific applications by convention
Ports help us distinguish between different applications on the same computer or server

- On private computers, port numbers can be random
- On public servers, port numbers should be constant and well-known (so users can access the right port)

IP Header: send to: 1.2.3.4
TCP Header: send to: port 80
HTTP: GET “Remember the milk!”
## Common Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>HTTP (Web)</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS (E2E encrypted Web)</td>
</tr>
<tr>
<td>25</td>
<td>SMTP</td>
</tr>
<tr>
<td>22</td>
<td>SSH</td>
</tr>
<tr>
<td>23</td>
<td>Telnet</td>
</tr>
<tr>
<td>53</td>
<td>DNS</td>
</tr>
</tbody>
</table>
Transmission Control Protocol (TCP)
The Two Generals Problem

- Two army divisions (blue) surround enemy (red)
  - Each division led by a general
  - Both must agree when to simultaneously attack
  - If either side attacks alone, defeat

- Generals can only communicate via messengers
  - Messengers may get captured (unreliable channel)
The Two Generals Problem

• How to coordinate?
  • Send messenger: “Attack at dawn”
  • What if messenger doesn’t make it?
The Two Generals Problem

• How to be sure messenger made it?
  • Send acknowledgment: “I delivered message”
In the “two generals problem”, can the two armies reliably coordinate their attack? (using what we just discussed)

• A. Yes (explain how)

• B. No (explain why not)
The Two Generals Problem

- Result
  - Can’t create perfect channel out of faulty one
  - Can only increase probability of success
Designing reliability over an unreliable link. What can go wrong?

A. Packets can be dropped
B. Packets can arrive out of order
C. Acknowledgements can arrive out of order
D. All of the above
E. There are more issues....
Designing reliability over an unreliable link. What can go wrong?

- **Problem: IP packets have a limited size. To send longer messages, we have to manually break messages into packets**
  - When sending packets: TCP will automatically split up messages
  - When receiving packets: TCP will automatically reassemble the packets
  - Now the user doesn’t need to manually split up messages!

- **Problem: Packets can arrive out of order**
  - When sending packets: TCP labels each byte of the message with increasing numbers
  - When receiving packets: TCP can use the numbers to rearrange bytes in the correct order

- **Problem: Packets can be dropped**
  - When receiving packets: TCP sends an extra message acknowledging that a packet has been received
  - When sending packets: If the acknowledgement doesn’t arrive, re-send the packet
Transmission Control Protocol (TCP)

- **Provides a byte stream abstraction**
  - Bytes go in one end of the stream at the source and come out at the other end at the destination
  - TCP automatically breaks streams into **segments**
- **Provides ordering**
  - Segments contain sequence numbers, so the destination can reassemble the stream in order
- **Provides reliability**
  - The destination sends acknowledgements (ACKs) for each sequence number received
  - If the source doesn’t receive the ACK, the source sends the packet again
- **Provides ports**
  - Multiple services can share the same IP address by using different ports