CS 161 SP19 Final Review

Day 1: Networking Review, DNS and DNSSEC

Networking Basics

Attacker Definitions
(for the purposes of this class)

- Man-in-the-middle (aka in-path): Attacker can see and modify traffic.
- On-path: Attacker can see but not modify traffic.
- Off-path: Attacker can’t see anything. (Still knows public information)

Remember all attackers can **send** whatever packets they want!

Think about what fields an attacker would need to guess correctly!

Protocols

A protocol is an **agreement on how to communicate**

- **Syntax**: How a communication is structured.
  - Format and order of how messages are sent and received.
- **Semantics**: What a communication means
  - Actions taken when transmitting, receiving, or timer expires

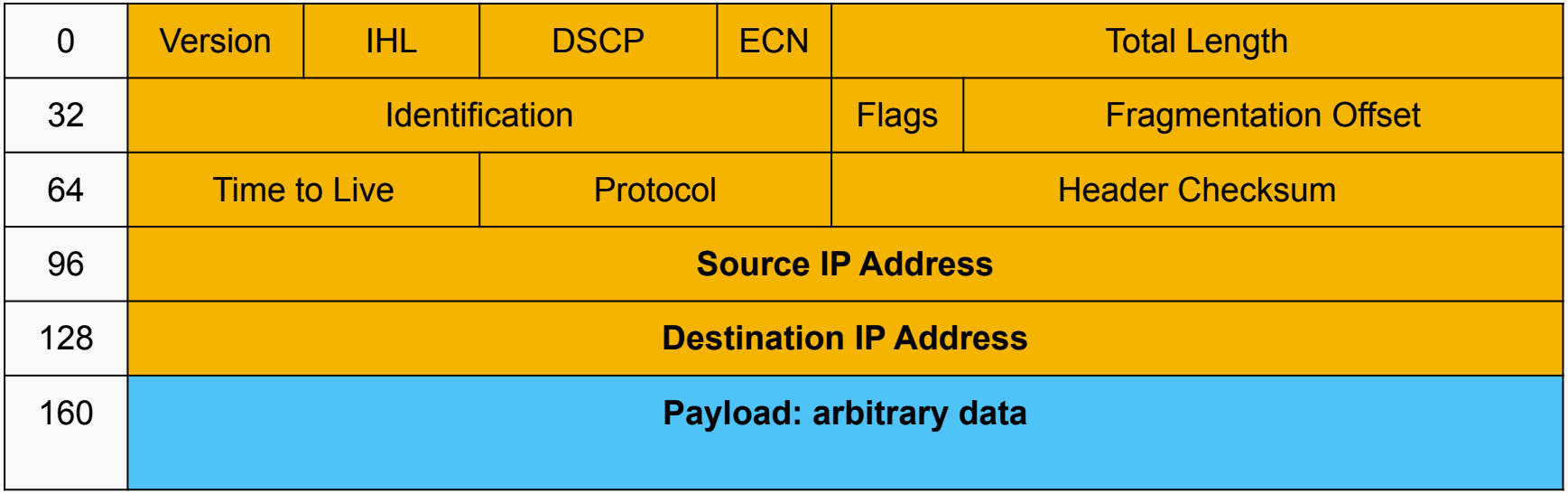
Examples: Wi-Fi, ARP, DHCP, TCP, UDP, IP, TLS, HTTP

Protocols: Example

Making a comment in lecture:

1. Raise your hand.
2. Wait to be called on.
3. Or: Wait for the speaker to **pause** and vocalize
4. If unrecognized (after **timeout**): Say “Excuse me”

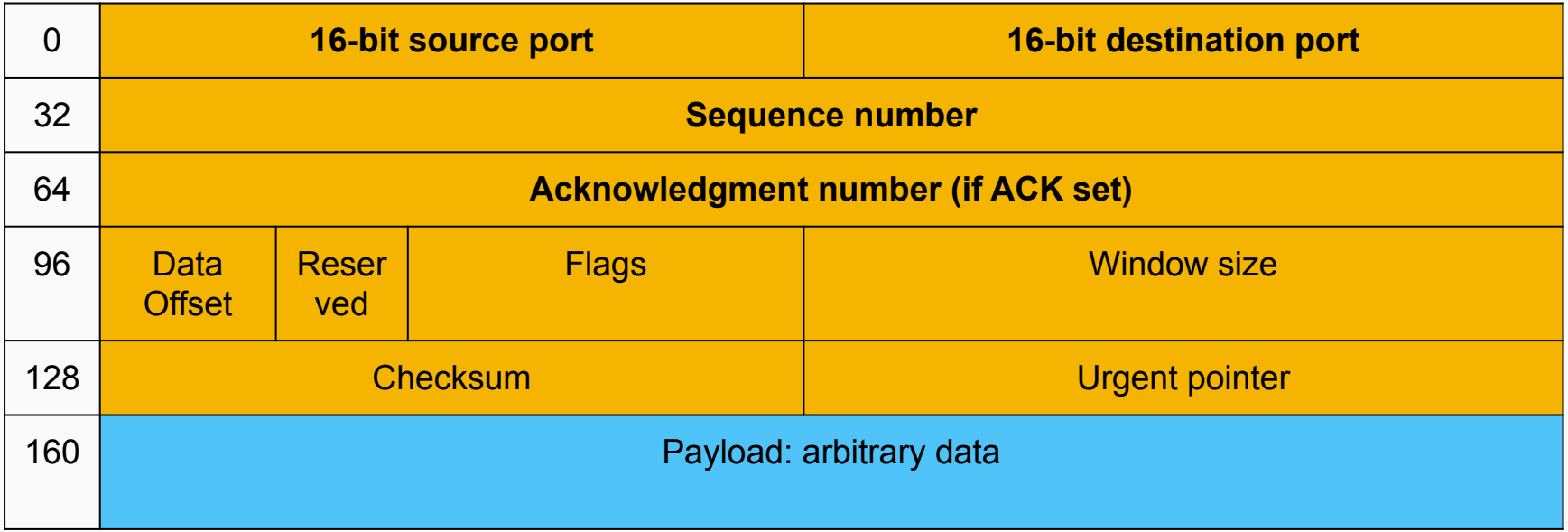
Headers: IP


0
Version
IHL
DSCP
ECN
Total Length
32
Identification
Flags
Fragmentation Offset
64
Time to Live
Protocol
Header Checksum
96
Source IP Address
128
Destination IP Address
160
Payload: arbitrary data

Headers: UDP

0	**16-bit source port**	**16-bit destination port**
32	16-bit length field	16-bit checksum
64	**Payload: arbitrary data**	

Headers: TCP


0
16-bit source port
16-bit destination port
32
Sequence number
64
Acknowledgment number (if ACK set)
96
Data Offset
Reserved
Flags
Window size
128
Checksum
Urgent pointer
160
Payload: arbitrary data

Headers: DNS

0	**Identification**							
16	QR	Opcode	AA	TC	RD	RA	(Just zeroes)	Response Code
32	QDCOUNT: Number of resource records in question section							
48	ANCOUNT: Number of resource records in answer section							
64	NSCOUNT: Number of resource records in authority section							
80	ARCOUNT: Number of resource records in authority section							
96	Arbitrary Data							

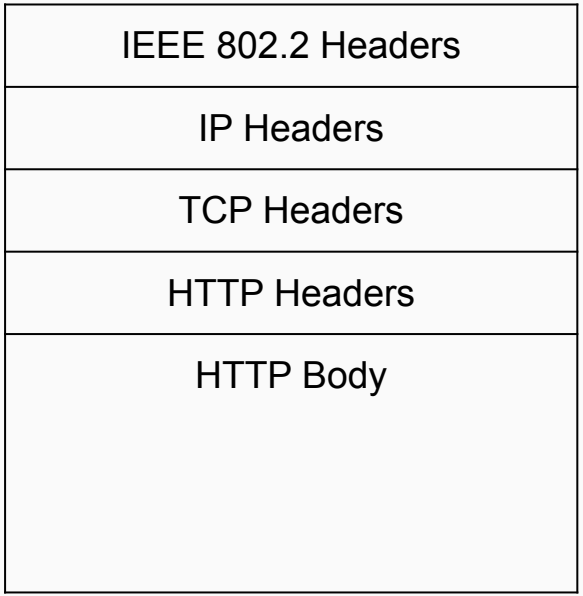
Layering

- The internet is like an onion: layers!
- Each layer relies on services provided by next layer below… and provides services to layer above it
- Similar to levels of abstraction when you program

Layering

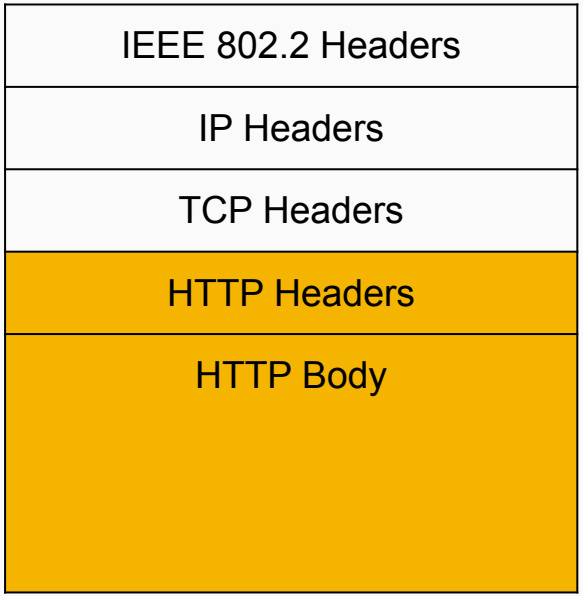
- **Layer 1**: Physical (bits on a wire, radio waves for WiFi, etc.)
- **Layer 2**: Link -- communication between "local" hosts
- **Layer 3**: InterLink -- communication between different "remote" hosts (IP)
- **Layer 4**: Transport -- TCP / UDP (more details soon)
- **Layers 5/6**: nobody cares.
- **Layer 7**: Application: things like HTTP (websites), SSH, etc.

If you receive an HTTP Packet it might look like this



What is the TCP payload?
What is the IP payload?

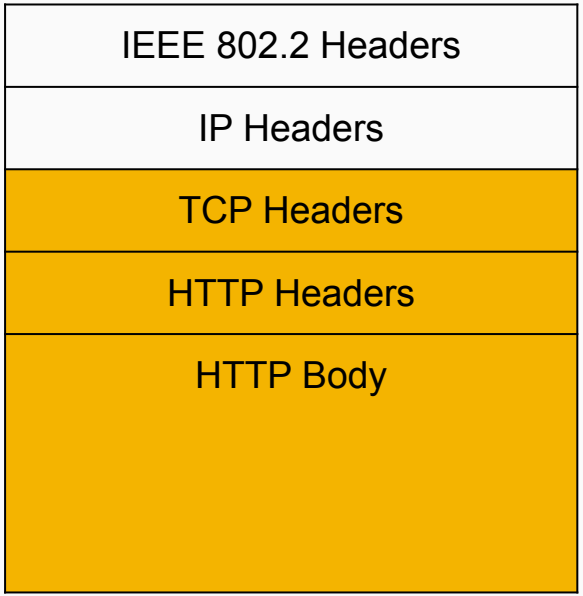
If you receive an HTTP Packet it might look like this



What is the TCP payload?

What is the IP payload?

If you receive an HTTP Packet it might look like this



What is the TCP payload?

What is the IP payload?

OSI Layer 1: Physical layer

- Bits on the wire(less)
- Copper cables, lasers, radio waves (standards: 802.11(a,b,g,n,ac), 802.3ae)
- line codes (e.g. 8b/10b, 64b/66b) for clock recovery
- You don’t need to worry about details in this layer.

OSI Layer 2: Link layer

- Defines communication between local hosts (think Wi-Fi network)
- Support for broadcast protocols
- Main protocol we care about here: ARP

ARP: Address Resolution Protocol

Converts IP addresses to MAC addresses.

- Host A wants to send a message to Host B
- Host A gets Host B's IP address (Call it X) from DNS
- Host A **broadcasts** "What is the MAC address of X?"
- Host B sends a message **only to A** "My IP is X and my MAC address is _".
- The IP, MAC pair gets cached

(If the X is outside the local network, then the gateway responds with their MAC address)

ARP: Spoofing

The ARP request is **public**. What happens if

- Alice sends out an ARP request for Bob’s MAC address
- Mallory responds with hers instead and beats Bob’s response.

ARP: Spoofing

Alice now believes Mallory’s MAC address is Bob’s and will send her packets intended for Bob.

OSI Layer 3: (Inter)network Layer

- Global routing
- No guarantee of reliability
- Packet delivery between different local networks, “best effort”

OSI Layer 4: Transport Layer

- Connection: one service (on a host) to another
- TCP: **Reliable**, in-order, connection-oriented transport
- UDP: **Unreliable**, datagram-based transport
  - A datagram is a single packet message

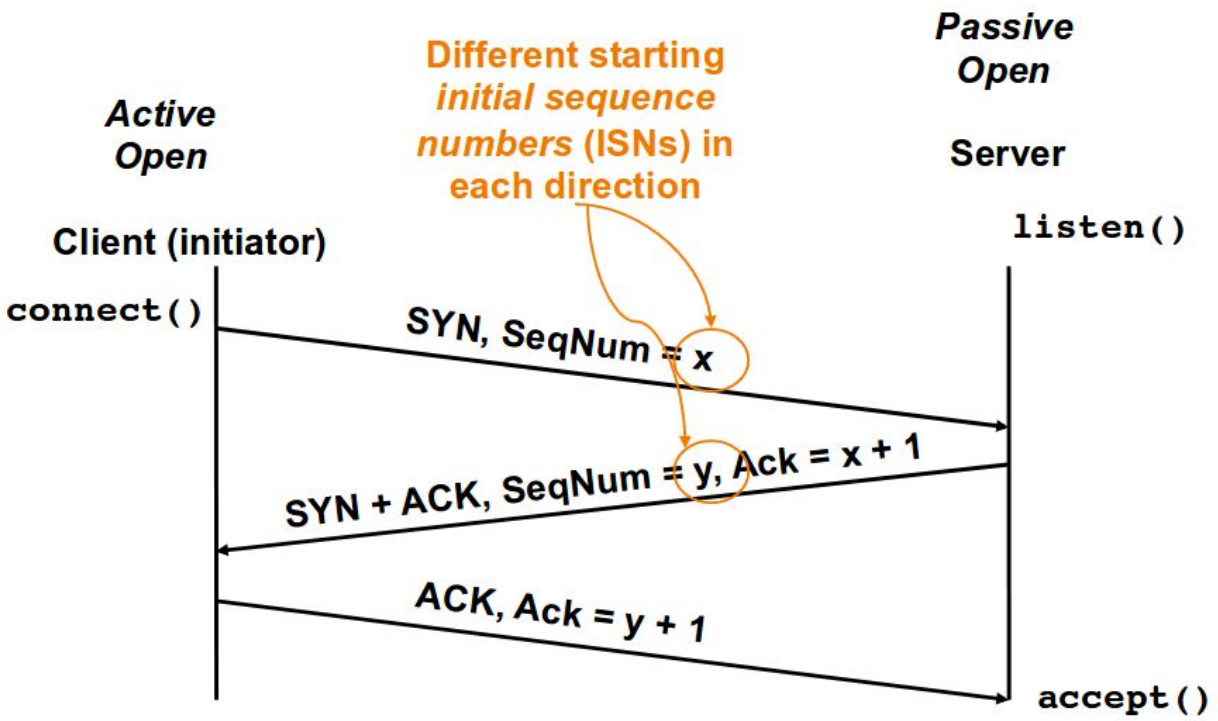
TCP: Transmission Control Protocol

Three-way handshake:

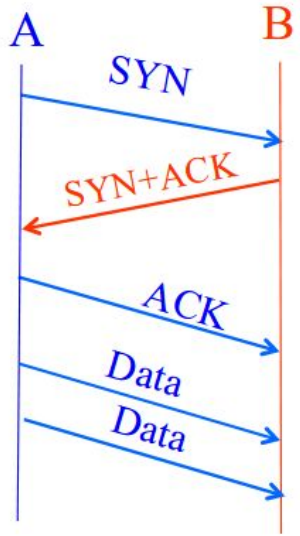
- Client sends: SYN, SEQ= x (arbitrary by the client)
- Server sends: SYN-ACK, SEQ= y (arbitrary by the server), ACK = x + 1
- Client sends: ACK, ACK = y + 1

The SEQ and ACK values swap depending on who’s sending the message. (You ACK the SEQ you receive.)

Timing Diagram: 3-Way Handshaking



Establishing a TCP Connection



Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

- Three-way handshake to establish connection
  - Host A sends a **SYN** (open; "synchronize sequence numbers") to host B
  - Host B returns a SYN acknowledgment (**SYN+ACK**)
  - Host A sends an **ACK** to acknowledge the SYN+ACK

TCP: Vulnerabilities

- TCP is not inherently secure
  - No confidentiality or integrity of data by default
- A malicious entity who knows the sequence numbers, port numbers, and IP addresses can spoof a connection
  - This means on-path attackers can easily inject data.
  - If an off-path attacker can guess this information, they can also inject data
- It also does not provide *availability*
  - TCP RST injection lets an attacker terminate connections

OSI Layer 7: Application Layer

- Application-to-application connection
- DHCP, DNS, TLS, etc.

DHCP: Dynamic Host Configuration Protocol

When you first connect to a network, you need a few things:

- An IP address
- The DNS server’s IP address
- The gateway’s (router's) IP address
- [Not really important for 161: Netmask]

DHCP: Handshake

Client Discover: “I need configuration information!”

Server Offer: “Here’s some possible configurations: A, B, C!”

Client Request: “I want to use configuration C!”

Server Acknowledge: “Okay, I have given you config C!”

This is all broadcast across the local network

DHCP: Spoofing

What happens when Alice accepts Mallory's configuration?

Mallory now controls Alice's:

- Gateway
- DNS (we talk about this later)

What can she do with this?

True/False Answers

If a laptop joining a WIFI network uses both DHCP and DNS, it will first use DHCP before using DNS.

When establishing a TCP connection, the client and the server engage in a three way handshake to determine the shared ISN they will both use for that connection.

.

Hosts that use DHCP on a wired networking technology such as Ethernet are protected against possible DHCP spoofing attacks.

ISPs are obligated to verify the IP source address on any traffic entering their network.

It’s difficult for an off-path attacker sending IP packets with a spoofed source to view the responses to those packets.

True/False Answers

If a laptop joining a WIFI network uses both DHCP and DNS, it will first use DHCP before using DNS. **True**

When establishing a TCP connection, the client and the server engage in a three way handshake to determine the shared ISN they will both use for that connection. **False.**

Hosts that use DHCP on a wired networking technology such as Ethernet are protected against possible DHCP spoofing attacks. **False.**

ISPs are obligated to verify the IP source address on any traffic entering their network. **False.**

It’s difficult for an off-path attacker sending IP packets with a spoofed source to view the responses to those packets. **True.**

Questions on Networking Basics?

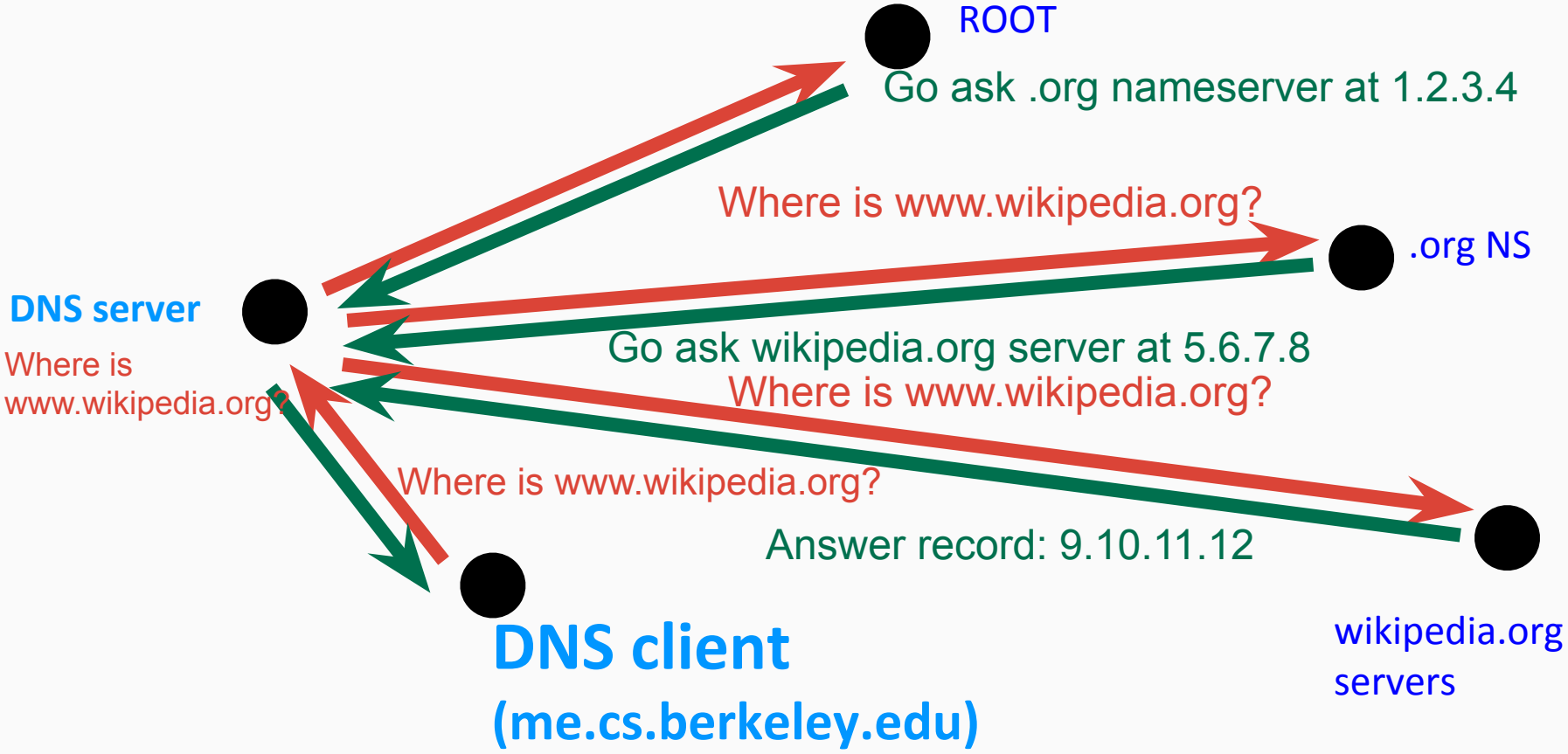
DNS

DNS: Domain Name System

Protocol for translating human-friendly domain names to a numerical address: www.berkeley.edu -> 52.10.99.247

Nameservers at least responsible for pointing to you in the right direction: .edu's name server might not know how to get to www.berkeley.edu, but it knows that berkeley.edu's name server knows and that name server's IP address.

DNS: Example (borrowed from cs168)



DNS: Resource Record (RR) Types

Each RR has a type and TTL (How long the record should be cached)

- A or AAAA: Matches a name w/ an IPv4 or IPv6 address

```
berkeley.edu      600     A       35.163.72.93
```

- NS: Gives you a name server responsible for a domain

```
edu.              172800    NS    a.edu-servers.net
```

A typical DNS response for: “I don’t know, but I can tell you who does and where they are”

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53577
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;berkeley.edu.                  IN      A

;; ANSWER SECTION:

;; AUTHORITY SECTION:
edu.                    172800  IN      NS      a.edu-servers.net.
edu.                    172800  IN      NS      b.edu-servers.net.

;; ADDITIONAL SECTION:
a.edu-servers.net.      172800  IN      A       192.5.6.30
b.edu-servers.net.      172800  IN      A       192.33.14.30
```

(What domain was queried for)

(What the address of the queried domain is. Note the lack of value)

(What the name servers responsible for .edu are)

(Possibly helpful additional information: Where the nameservers are)

A typical DNS response for: “Hey I found what you’re asking for”

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39459
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;berkeley.edu.                  IN      A

;; ANSWER SECTION:
berkeley.edu.           16      IN      A       35.163.72.93

;; AUTHORITY SECTION:
berkeley.edu.           43860   IN      NS      adns2.berkeley.edu.
berkeley.edu.           43860   IN      NS      adns1.berkeley.edu.

;; ADDITIONAL SECTION:
adns1.berkeley.edu.     90569   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     693     IN      A       128.32.136.14
adns2.berkeley.edu.     1598    IN      AAAA    2607:f140:ffff:fffe::e
```

(What domain was queried for)

(What the address of the queried domain is)

(What the name servers responsible for berkeley.edu are)

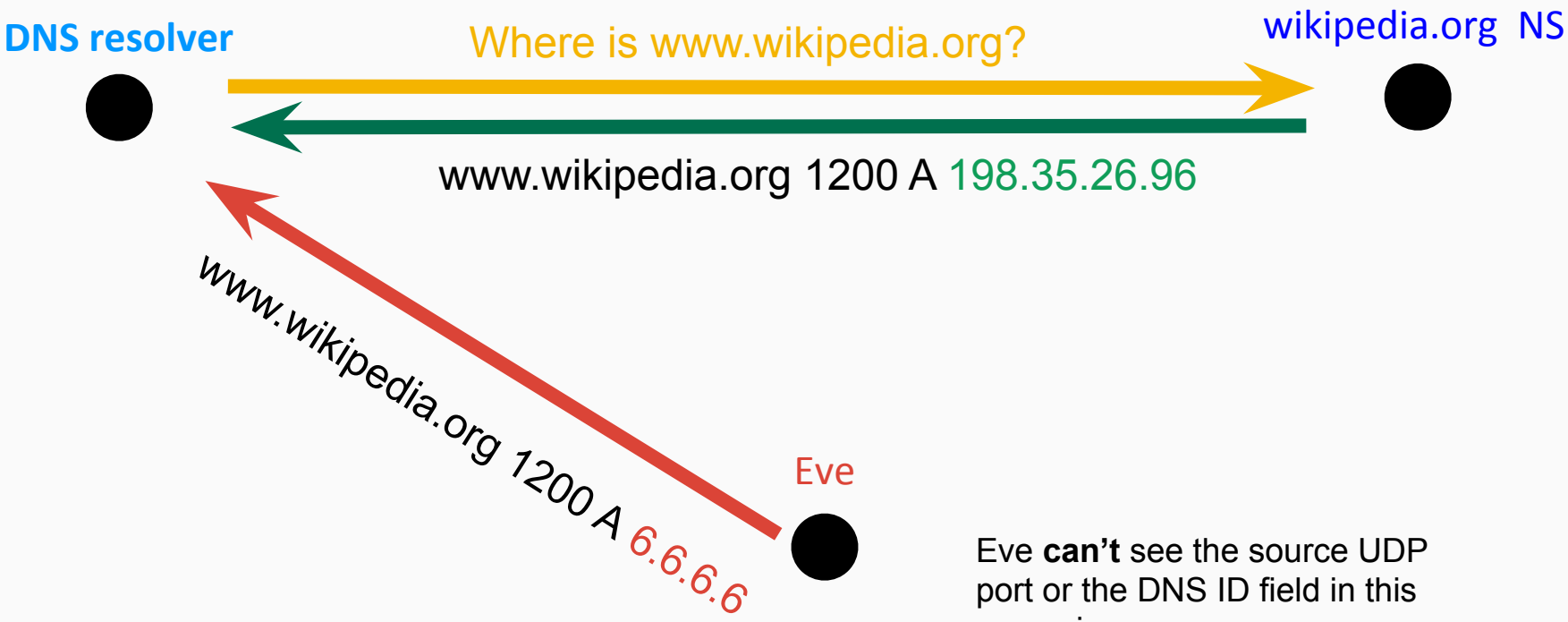
(Possibly helpful additional information: Where the nameservers are)

DNS: Vulnerabilities

- Malicious DNS server
  - .edu name server telling us berkeley.edu's name server is ns1.stanford.edu's IP address
    - Why doesn't bailiwick protect against this?
- On-path eavesdropper
  - Spoofing
- Off-path attacker
  - Blind-spoofing

What do spoofers need to guess correctly to be accepted by the victim?

DNS: Off-path spoofing



Eve **can't** see the source UDP port or the DNS ID field in this scenario.

DNS: Kaminsky Attack

- Once a resolver receives a response, it will **cache** the results. So not only does the spoofer need to beat the real result, once it loses, it can't try again for a bit.
- Kaminsky's attack subverts this by trying to poison the name server's IP address in the Additional section.

DNS: Kaminsky attack

If this attack succeeds, then any future query for any domain in wikipedia.org will be sent to Eve.

If it doesn't, it doesn't matter. The cache has not been altered. Just ask on another random subdomain.

Questions on DNS?

True/False Questions

Source port randomization helps defend against an off-path attacker performing the Kaminsky DNS cache poisoning attack.

“Bailiwick” checks in modern DNS resolvers will prevent a malicious name server responsible for foo.com from using the Additional fields in its DNS responses to poison cache entries for bar.com.

In the event where the domain name to IP address binding changes, the DNS server responsible for the given domain name sends invalidation messages to clients in order to flush their mappings.

Randomizing the DNS query identifier prevents an on-path attacker from spoofing DNS responses.

True/False Answers

Source port randomization helps defend against an off-path attacker performing the Kaminsky DNS cache poisoning attack. **True.**

“Bailiwick” checks in modern DNS resolvers will prevent a malicious name server responsible for foo.com from using the Additional fields in its DNS responses to poison cache entries for bar.com. **True.**

In the event where the domain name to IP address binding changes, the DNS server responsible for the given domain name sends invalidation messages to clients in order to flush their mappings. **False.**

Randomizing the DNS query identifier prevents an on-path attacker from spoofing DNS responses. **False.**

Previous Exam Questions

A. No additional capabilities
B. A $10k GPU compute cluster
C. Can compromise Sarah's home router
D. Can compromise the primary nameserver for the zone

(a) In the following table, **mark with an X** which attacks each adversary could successfully perform against Sarah if everyone uses **DNS without DNSSEC**:

	A	B	C	D
Spoof existence of records that actually don't exist in the zone				
Spoof values of records that exist in the zone				
Spoof non-existence of records that actually do exist in the zone				
Enumerate all names in the zone				

Previous Exam Questions

A. No additional capabilities
B. A $10k GPU compute cluster
C. Can compromise Sarah's home router
D. Can compromise the primary nameserver for the zone

(a) In the following table, **mark with an X** which attacks each adversary could successfully perform against Sarah if everyone uses **DNS without DNSSEC**:

	A	B	C	D
Spoof existence of records that actually don't exist in the zone	X	X	X	X
Spoof values of records that exist in the zone	X	X	X	X
Spoof non-existence of records that actually do exist in the zone	X	X	X	X
Enumerate all names in the zone				X

DNSSEC

Trust Delegation without DNSSEC

- Everyone has root's [.] nameserver IP hardcoded
- Remember requesting `www.berkeley.edu`:
  - Ask root
  - Root gives an answer:
    - `edu. NS a.edu-servers.net.`
      `a.edu-servers.net. A 192.5.6.30`
  - "Delegated" trust to .edu's nameservers
  - Repeat recursively
- **Problem**: no integrity, so response may have been modified

Why DNSSEC?

DNS provides poor security, so DNSSEC was introduced as an alternative, guaranteeing these properties:

- **Integrity** (not confidentiality) via signatures
- Cache-ability of the signatures
- Extends and remains backwards compatible with DNS

Key Idea: PKI

- DNSSEC provides a public-key infrastructure (PKI)
- Everyone has root's [.] nameserver IP and Key Signing Key (KSK)
- Remember requesting `www.berkeley.edu`:
  - `edu. NS a.edu-servers.net.`
  - `a.edu-servers.net. A 192.5.6.30`
  - `. DNSKEY <my Zone Signing Key (ZSK)>`
  - `. DNSKEY <my Key Signing Key (KSK)>`
  - `. RRSIG DNSKEY <SIGN(all DNSKEY records, my KSK)>`
  - `edu. DS <HASH(.edu's KSK)>`
  - `edu. RRSIG DS <SIGN(all DS records, my ZSK)>`
  - `. RRSIG NS <SIGN(all NS records, my ZSK)> [technically unnecessary]`
  - `. RRSIG A <SIGN(all A records, my ZSK)> [technically unnecessary]`

So in (Long) Summary

- If you trust my Key Signing Key (KSK)...
- I can give you my Zone Signing Key (ZSK) and KSK in DNSKEY
  - RRSIG DNSKEY contains signature on DNSKEY records with my KSK
  - So you now trust my ZSK
- I can give you the answer to your query (typically A and NS records)
  - RRSIG A and RRSIG NS are signatures on these records with my ZSK
- Finally, I give you a DS record (delegated signer)
  - DS contains a hash of my child zone's KSK
  - RRSIG DS contains signature on DS with my ZSK
  - So you now know what my child's KSK is
- If you trust me, I can **answer** your query and **delegate trust**

Practice Question

- Say you make a request for www.berkeley.edu's A record (IPv4 address), which is answered by .berkeley.edu's authoritative nameserver
- [Assume all caches empty.]
- While doing so, what nameservers do you contact?
- Which of the following record types do you need from each nameserver?
  - DNSKEY from:
  - DS records from:
  - RRSIG records (specify what records the RRSIGs sign):

Practice Answer

- Say you make a request for www.berkeley.edu's A record (IPv4 address), which is answered by .berkeley.edu's authoritative nameserver
- [Assume all caches empty.]
- While doing so, what nameservers do you contact?
  - Authoritative nameservers for . (root), .edu, .berkeley.edu
- Which of the following record types do you get from each nameserver?
  - DNSKEY from: . (root), .edu, .berkeley.edu
  - DS records from: . (root), .edu
  - RRSIG records (specify what records each RRSIG signs): RRSIG DNSKEY from root, .edu, .berkeley.edu. RRSIG DS from root, .edu. RRSIG A from root, .edu, .berkeley.edu. RRSIG NS from root, .edu. (A/NS RRSIGs for intermediates are technically unnecessary.)

Practice Question

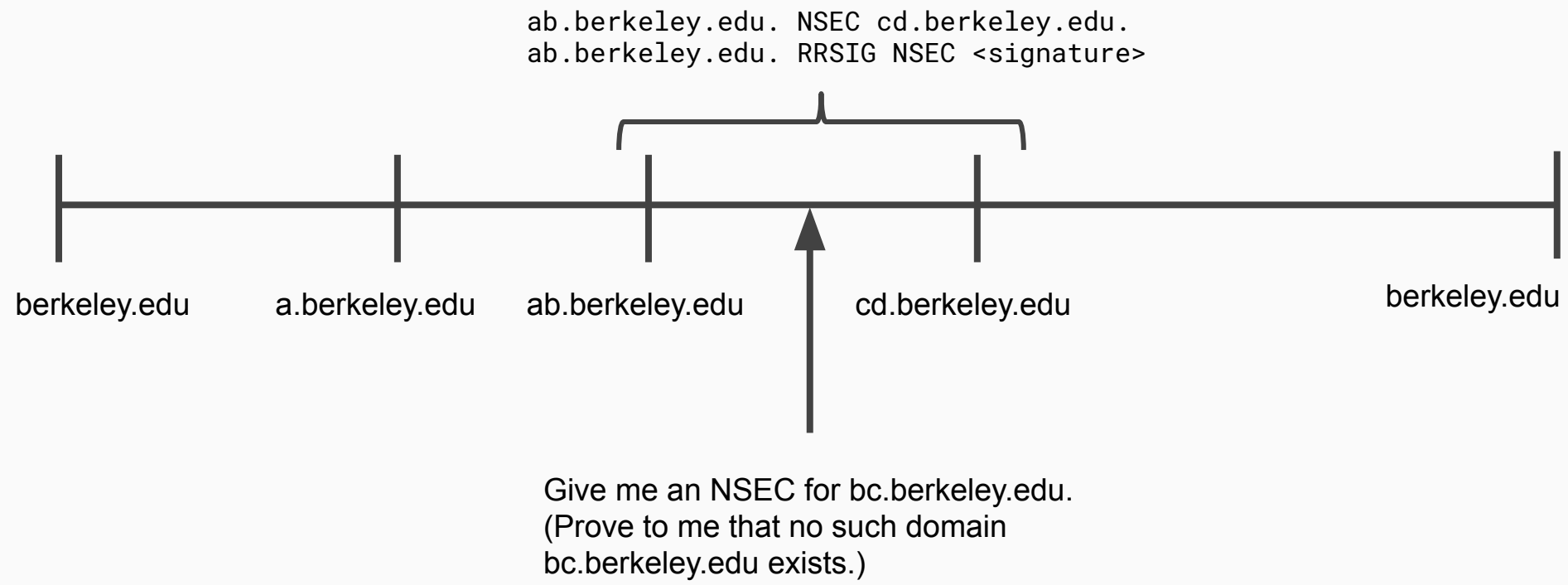
- Does DNSSEC allow for…
  - changing one's ZSK without contacting the parent nameserver?
  - changing one's KSK without contacting the parent nameserver?
  - changing one's A records without contacting the parent nameserver?
- Explain what (cryptographic) operations you need to do in order to update each of these.

Practice Answer

- Does DNSSEC allow for…
  - changing one's ZSK without contacting the parent nameserver? Yes.
    - You will need to update all RRSIGs on all records, and use your KSK private-key to update the RRSIG on DNSKEY records.
  - changing one's KSK without contacting the parent nameserver? No.
    - Parent's DS record will be wrong (hash of your KSK public-key).
    - Ask parent to update their DS record to the new hash. Then use your new KSK private-key to sign a new RRSIG for DNSKEY records.
  - changing one's A records without contacting the parent nameserver? Yes.
    - You will need to update RRSIG on A records by signing with your ZSK private-key.

NSEC

- Problem: how to assert a domain DOES NOT exist?
  - We could sign: "Domain X does not exist" using our ZSK
  - But that would require online cryptography ⇒ added latency
- NSEC provides *offline* authenticated denial to solve this
  - Can do all the crypto "in advance"


ab.berkeley.edu. NSEC cd.berkeley.edu.
ab.berkeley.edu. RRSIG NSEC <signature>
berkeley.edu
a.berkeley.edu
ab.berkeley.edu
cd.berkeley.edu
berkeley.edu
Give me an NSEC for bc.berkeley.edu.
(Prove to me that no such domain
bc.berkeley.edu exists.)

Enumerating NSEC

- [Perceived] Problem with NSEC: it allows an attacker to discover all subdomains of a site
  - Keep making requests for invalid domain names, and collect all NSEC records
  - Faster than trying to guess valid subdomains and making DNS requests for them
- NSEC**3** tries to solve this by using hashes
  - Attest that there are no domains whose HASH *H* falls between H(N) and H(M)
  - Not very effective against modern GPUs which can make many hash tries per second

Questions on DNSSEC?

Practice Question

A. No additional capabilities
B. A $10k GPU compute cluster
C. Can compromise Sarah's home router
D. Can compromise the primary nameserver for the zone
E. Can compromise the primary nameserver and offline signing key for the zone

(b) Remember that when using DNSSEC, the zone is signed with a key that is not stored on the primary nameserver. Instead, the key is stored at an "offline" location. Also, NSEC records sign each adjacent pair of names in the zone. In the following table, **mark with an X** which attacks each adversary could successfully perform against **DNSSEC with NSEC records**:

	A	B	C	D	E
Spoof existence of records that actually don't exist in the zone					
Spoof values of records that exist in the zone					
Spoof non-existence of records that actually do exist in the zone					
Enumerate all names in the zone					

Practice Question

A. No additional capabilities
B. A $10k GPU compute cluster
C. Can compromise Sarah's home router
D. Can compromise the primary nameserver for the zone
E. Can compromise the primary nameserver and offline signing key for the zone

(b) Remember that when using DNSSEC, the zone is signed with a key that is not stored on the primary nameserver. Instead, the key is stored at an "offline" location. Also, NSEC records sign each adjacent pair of names in the zone. In the following table, **mark with an X** which attacks each adversary could successfully perform against **DNSSEC with NSEC records**:

	A	B	C	D	E
Spoof existence of records that actually don't exist in the zone					X
Spoof values of records that exist in the zone					X
Spoof non-existence of records that actually do exist in the zone					X
Enumerate all names in the zone	X	X	X	X	X