CS 161: Cats and Penguins Midterm 2 Review

4/3/18
Kevyn, Karthik

Slides made by Kevyn, Karthik, Sam, Chris, and Paul with material “borrowed” from Dave Wagner, Nick Weaver, Raluca Ada Popa, and Scott Shenker.

Topics

- RSA signatures (Karthik)
- Networking (Kevyn)
- TLS (Karthik)
- Web (Rip, no time)
- Good luck?

Our format: Topic review, then questions

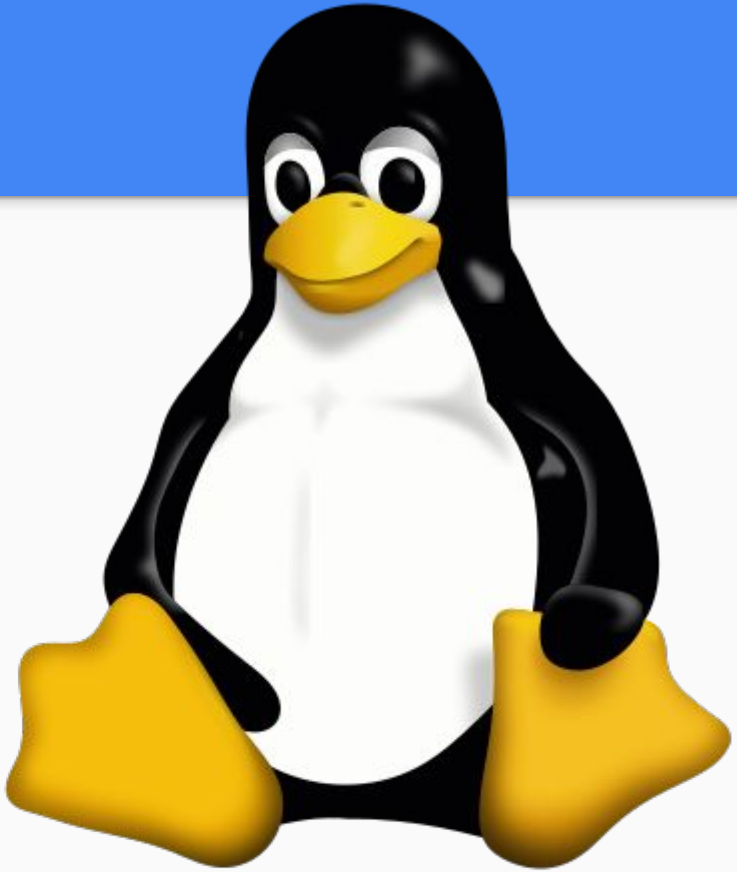
Disclaimer

This review is meant to help spot check your understanding of the course material and hopefully direct your attention to things you have overlooked in your studying.

This product has not been evaluated by the FDA and is not approved to diagnose, treat, cure, or prevent disease.

There could be stuff on the exam that is not covered in the review, and stuff in the review is not necessarily in the exam.

Tux and Broccoli



RSA Signatures

- The RSA signature is MAC's asymmetric counterpart
  - Messages are signed with a private key and verified with a public key
- Generating Signature
  - 2 large primes, **p** and **q**, are chosen and their product, **n**, is computed. n is the public key
  - From p and q, a private key, **d**, is derived
  - The message, M, is hashed to produce H(M)
  - The signature, S, is produced as follows: $S = H(M)^d \bmod n$
- Verifying Signature
  - Received message is decrypted (if encrypted) and hashed to produce H(M)
  - If $H(M) == S^3 \bmod n$, signature is correct
- For a more mathematically rigorous treatment of RSA signatures, refer to https://inst.eecs.berkeley.edu/~cs161/fa17/notes/Signatures.2.28.pdf

RSA Signatures

- RSA signatures ensure the following:
    - Even if a malicious party is allowed to choose messages to be signed, they are unable to guess what the signature would be on a never before seen message
- RSA signatures provide integrity
    - Verifiable signatures are impossible to produce without the private key. As a result, attackers cannot tamper with message and produce new, valid signature.
- RSA signatures provide authentication
    - Only the owner of the private key can produce verifiable signatures.
- RSA signatures provide non-repudiation
    - Proof is easy to provide to a third party, since anyone can verify a signature with the signer's public key

RSA Signatures

- What if we skipped the hashing step?
  - $S = M^d \bmod n$
  - If $M == S^3 \bmod n$, signature is correct

RSA Signatures

- What if we skipped the hashing step?
  - $S = M^d \bmod n$
  - If $M == S^3 \bmod n$, signature is correct
- You can forge signatures
  - Pick a random S, compute $M = S^3 \bmod n$, S is now a valid signature of M!
- Attacker has no control over M
  - By making M a cryptographic hash, attackers cannot find messages that hash to M

Password Hashing

- Problem: If a website leaks their password database, then their user accounts are compromised
  - Attackers can use the leaked passwords to login as other users
- Idea: store data that can be used to verify a password, but cannot (easily) be used to compute a valid password

Password Hashing

Joe runs a large website that allows users to log in and share images. When a new user sets up their account, the website hashes their password with **SHA256** and stores the hash in a database.

When a user logs in, the website hashes the supplied password with SHA256 and compares it to the stored hash. Joe figures that with this scheme, if anyone hacks into your database they will only see hashes and won’t learn your users’ passwords.

Out of curiosity, Joe does a Google search on several hashes in the database and is alarmed to find that, for a few of them, the Google search results reveal the *corresponding password*. He comes to you for help.

Password Hashing Questions

1. What mistake did Joe make in how he stored passwords?
2. What is the consequence of this mistake? In other words, what is the risk that it introduces and how many of Joe's users could be affected? Does it affect only users whose password hashes are available in Google search, or does it go beyond that?
3. How should Joe store passwords? More specifically, if a user's password is w, what should Joe store in the database record for that user?

Password Hashing Questions

1. What mistake did Joe make in how he stored passwords?

He didn’t use a salt.
(His other mistake was to use a hash that is too fast, though that doesn’t really explain why the hash turned up in a Google search, so this didn’t receive full credit.)

Password Hashing Questions

2. What is the consequence of this mistake? In other words, what is the risk that it introduces and how many of Joe's users could be affected? Does it affect only users whose password hashes are available in Google search, or does it go beyond that?

If the database is leaked (e.g., server compromise), the attacker can mount offline password guessing attacks. Such an attacker might be able to recover many of the users' passwords—not just those whose password hashes are listed in Google search.

Password Hashing Questions

3. How should Joe store passwords? More specifically, if a user's password is w, what should Joe store in the database record for that user?

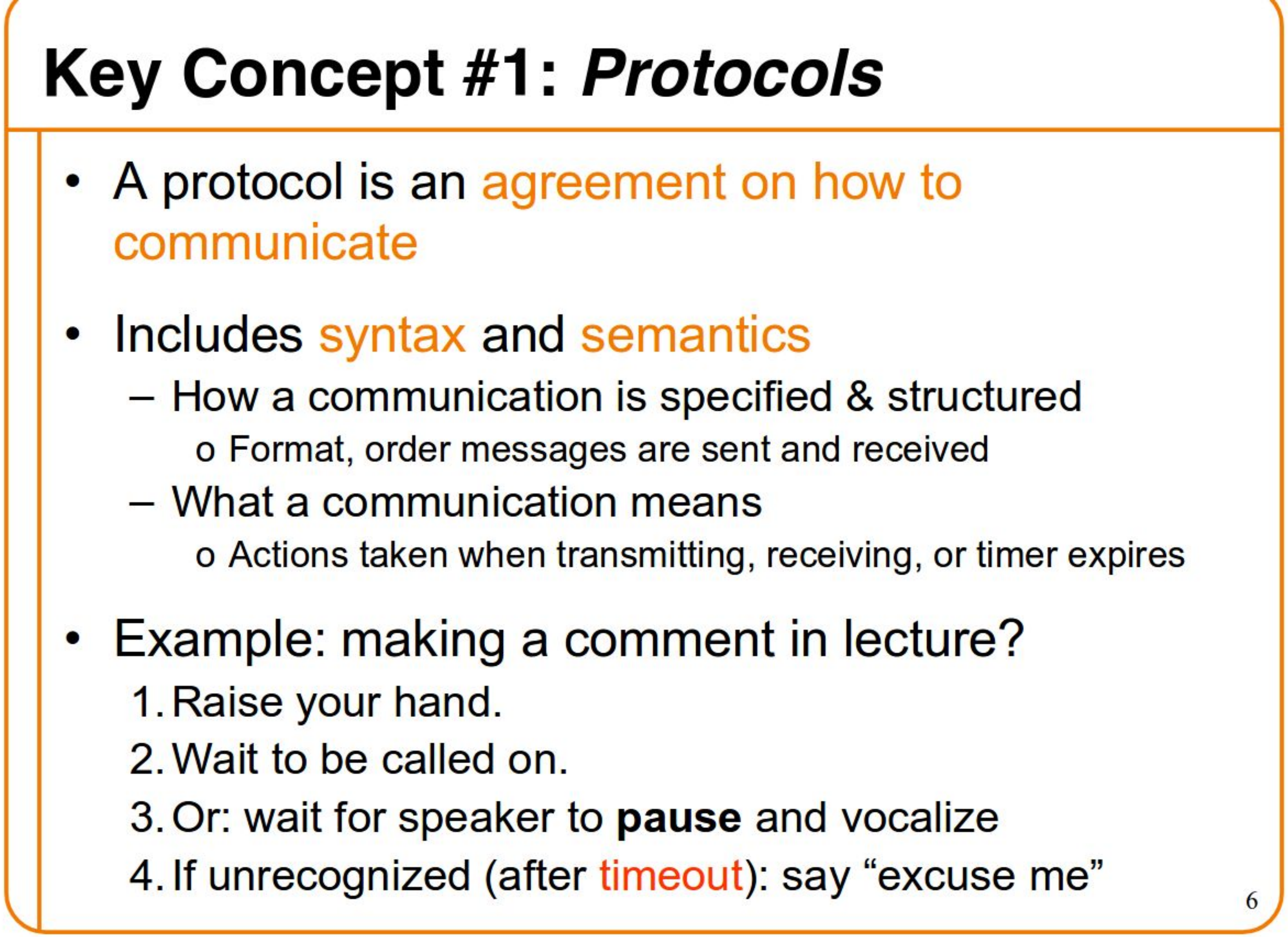
s,F(w,s) where s is a random salt chosen independently for each user and where F is a slow cryptographic hash, e.g., SHA256 iterated many times ($F(x) = H(H(\cdots(x)\cdots))$ where H is SHA256).

Password Hashing Summary

- Hash passwords to help protect their confidentiality
- Use salts to prevent attackers from memorizing common password hashes
- Make the salts unique per user to prevent attackers from cracking multiple passwords at once
- Use a slow cryptographic hash function to make it harder for attackers to brute force a password (many iterations of a slow hash function works)
- Passwords can still be brute-forced/guessed after all these defenses!

Networking Overview: “Everything” you need to know, in 50 minutes

Just kidding, we can’t fit everything, and we don’t have 50 minutes to spend on this.



Key Concept #1: *Protocols*

- A protocol is an agreement on how to communicate
- Includes syntax and semantics
  - How a communication is specified & structured
    - Format, order messages are sent and received
  - What a communication means
    - Actions taken when transmitting, receiving, or timer expires
- Example: making a comment in lecture?
  1. Raise your hand.
  2. Wait to be called on.
  3. Or: wait for speaker to **pause** and vocalize
  4. If unrecognized (after timeout): say "excuse me"

6

Ethernet (the protocol)

802.11(Wifi)

ARP

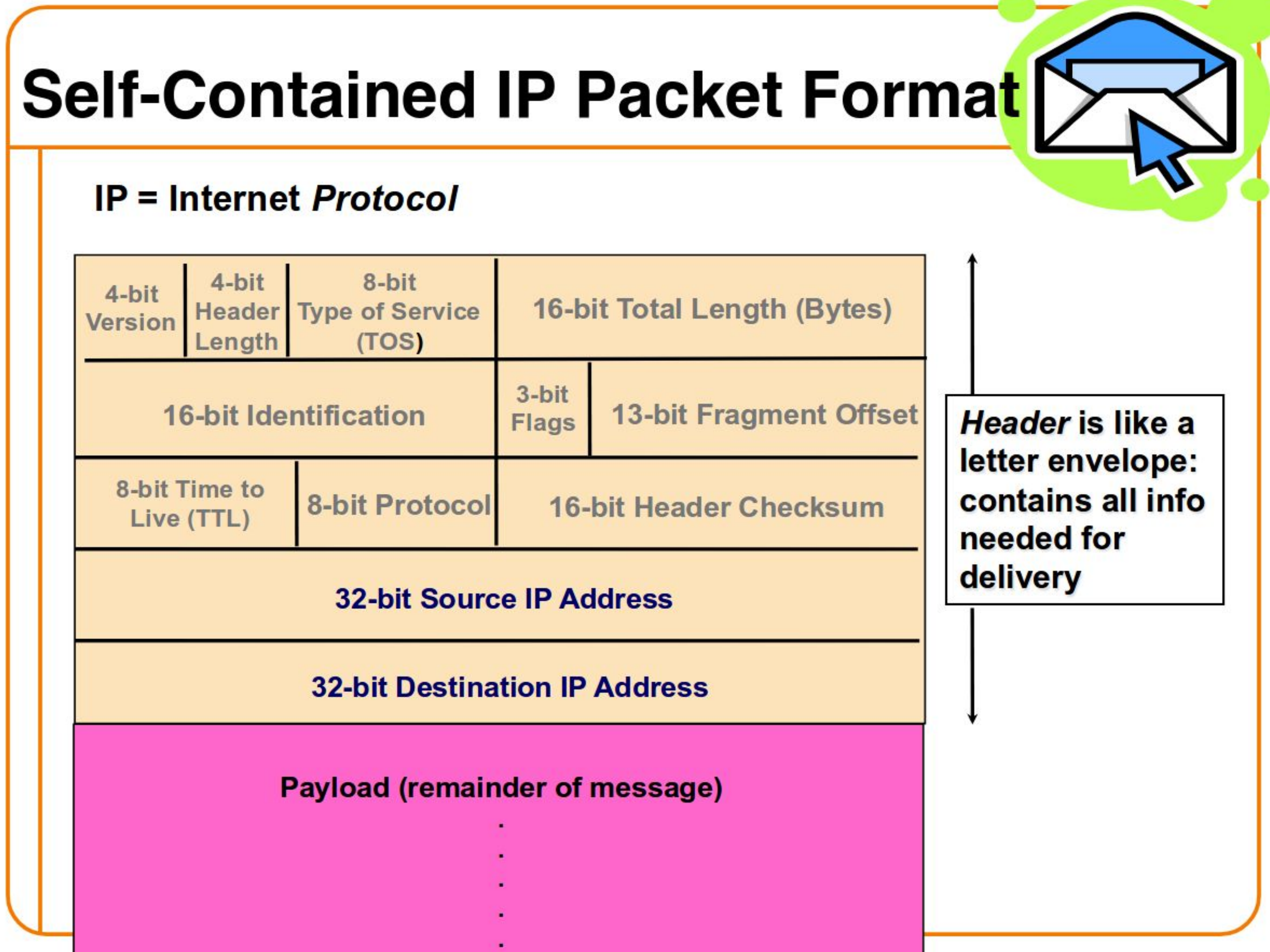
DHCP

TCP

UDP

IP

TLS



Self-Contained IP Packet Format

IP = Internet *Protocol*

4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)
16-bit Identification		3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address			
32-bit Destination IP Address			

Payload (remainder of message)
.
.
.
.
.

Header **is like a letter envelope: contains all info needed for delivery**

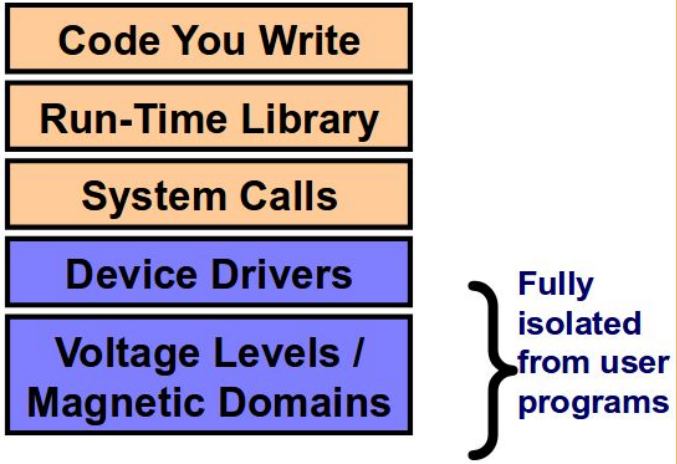
It might be nice to write this down and forget about it.

Also good to write down: TCP and UDP headers.

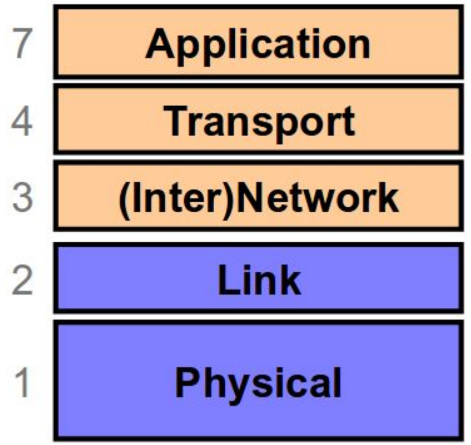
Available for purchase separately.

Key Concept #3: *Layering*

- Internet design is strongly partitioned into layers
  - Each layer relies on services provided by next layer below …
  - … and provides services to layer above it
- Analogy:
  - Consider structure of an application you've written and the "services" each layer relies on / provides


Code You Write
Run-Time Library
System Calls
Device Drivers
Voltage Levels / Magnetic Domains
Fully isolated from user programs

Internet Layering ("Protocol Stack")



Note on a point of potential confusion: these diagrams are always drawn with lower layers **below** higher layers …

But diagrams showing the layouts of packets are often the *opposite*, with the lower layers at the **top** since their headers precede those for higher layers

Important Things to remember:

7. TLS, DNS

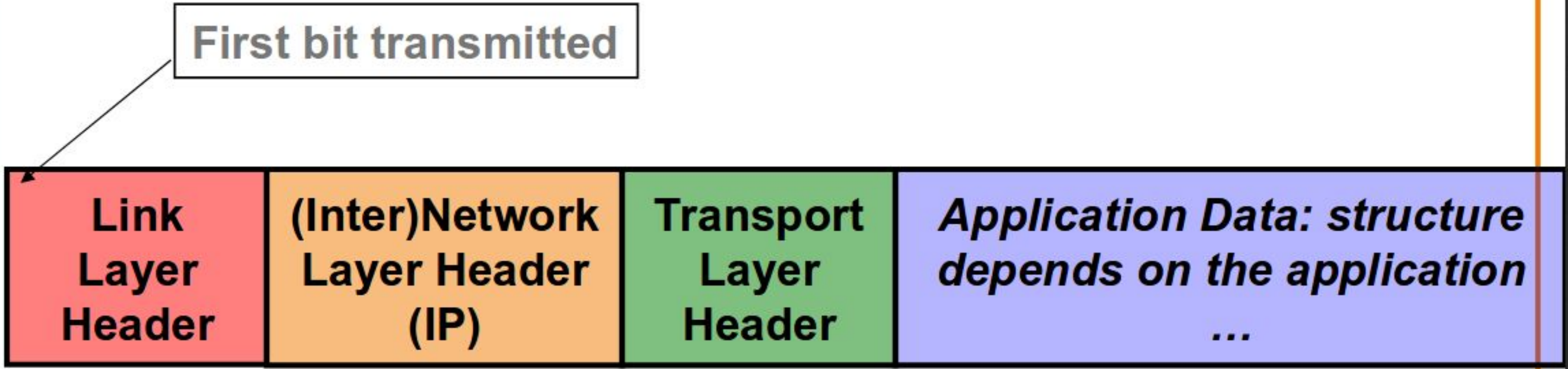
4. TCP, UDP

3. IP

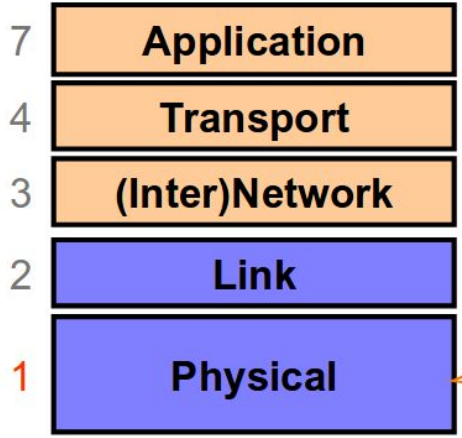
2. ARP

1. Bits on a "wire"(less)

Horizontal View of a Single Packet

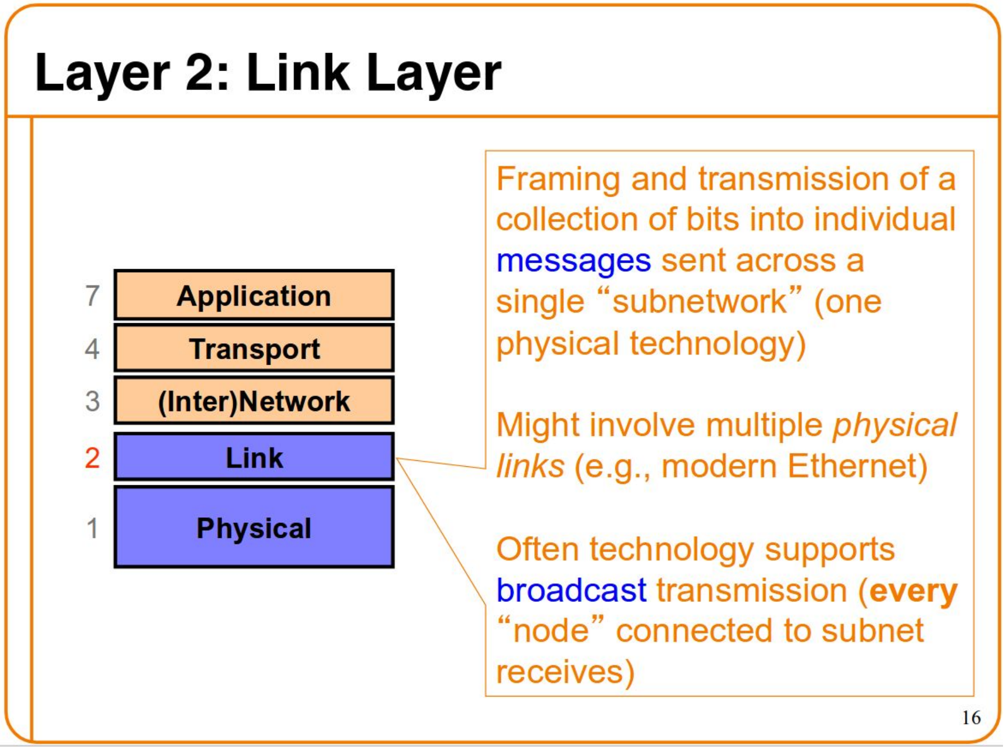

First bit transmitted
Link Layer Header
(Inter)Network Layer Header (IP)
Transport Layer Header
Application Data: structure depends on the application …

Layer 1: Physical Layer



Encoding bits to send them over a single **physical link**

e.g. patterns of *voltage levels / photon intensities / RF modulation*



Layer 2: Link Layer

7 Application
4 Transport
3 (Inter)Network
2 Link
1 Physical

Framing and transmission of a collection of bits into individual messages sent across a single "subnetwork" (one physical technology)

Might involve multiple *physical links* (e.g., modern Ethernet)

Often technology supports broadcast transmission (**every** "node" connected to subnet receives)

Do y'all know what subnets are?
Example:
Are 192.168.1.5 and 192.168.0.6 on the same subnet?
Subnet mask: 192.168.0.0 Yes
Subnet mask: 192.168.1.0 No

ARP: Address Resolution Protocol

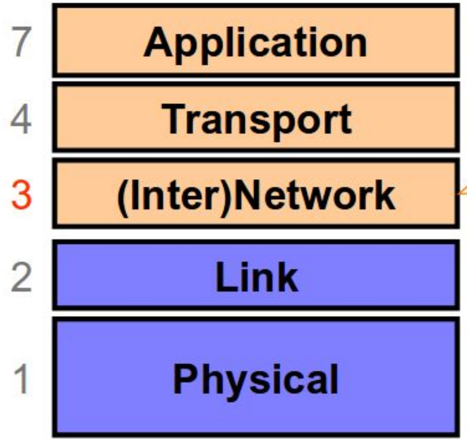
Basically, you need to know the MAC address for L2.

So you yell: Hey who has this IP address?

And hope the device responds.

Does anything seem dangerous here?

Layer 3: (Inter)Network Layer *(IP)*



Bridges multiple "subnets" to provide *end-to-end* internet connectivity between nodes

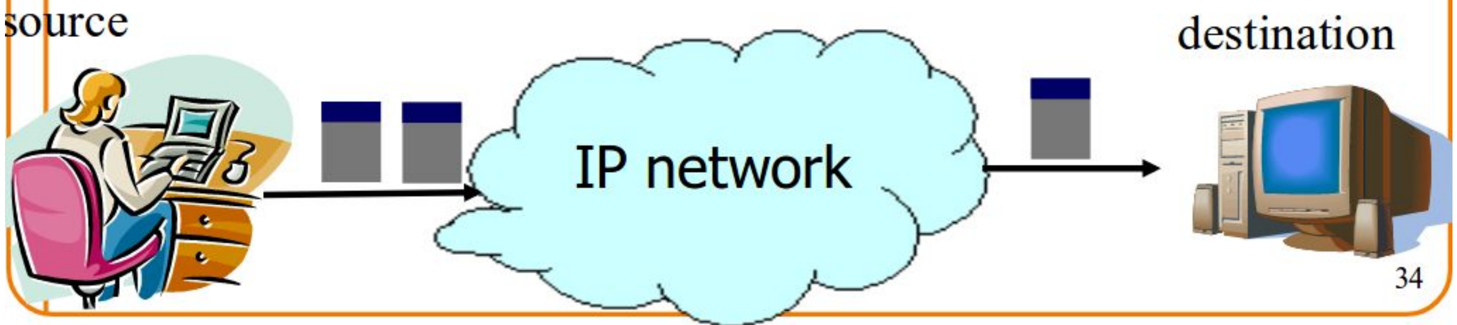
- Provides global addressing

Works across different link technologies

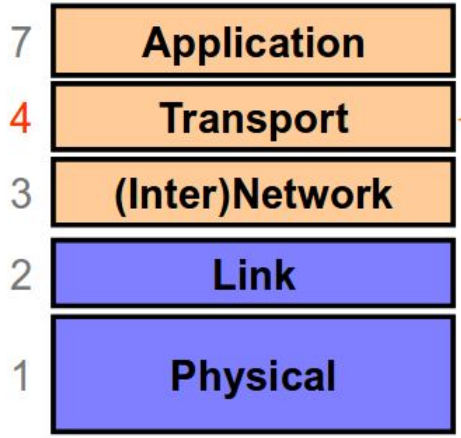
Different **for each Internet "hop"**

IP: "*Best Effort*" Packet Delivery

- Routers inspect destination address, locate "next hop" in forwarding table
  - Address = ~unique identifier/locator for the receiving host
- Only provides a "*I'll give it a try*" delivery service:
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



Layer 4: Transport Layer



End-to-end communication between processes

Different services provided:
TCP = reliable *byte stream*
UDP = unreliable *datagrams*

(Datagram = single packet message)

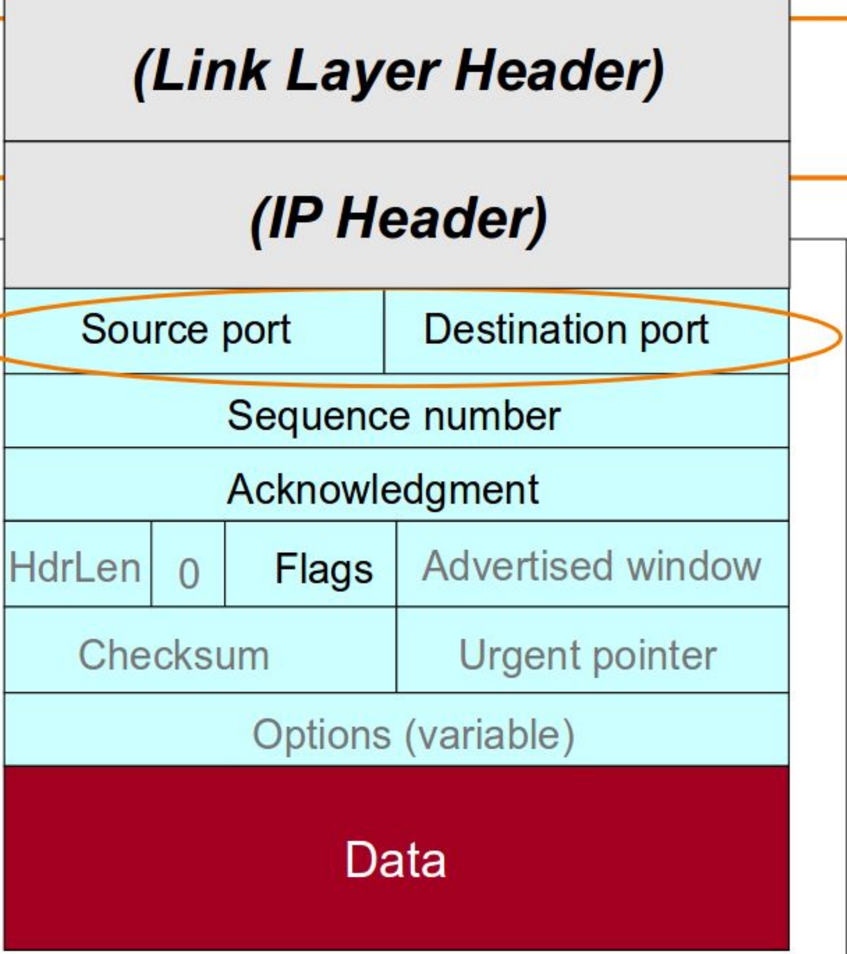
“Best Effort” is Lame! What to do?

- It’s the job of our Transport (layer 4) protocols to build services our apps need out of IP’s modest layer-3 service
- #1 workhorse: TCP (Transmission Control Protocol)
- Service provided by TCP:
  - Connection oriented (explicit set-up / tear-down)
    - End hosts (processes) can have multiple concurrent long-lived communication
  - **Reliable**, in-order, *byte-stream* delivery
    - Robust detection & retransmission of lost data

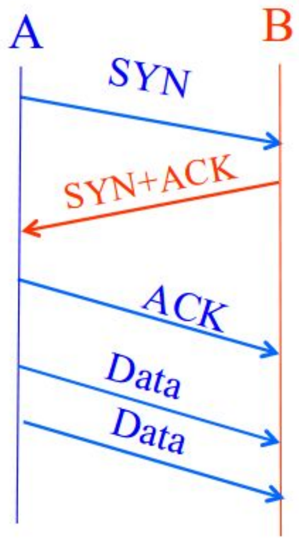
TCP Header

Ports are associated with OS processes

IP source & destination addresses plus TCP source and destination ports uniquely identifies a TCP connection



Establishing a TCP Connection

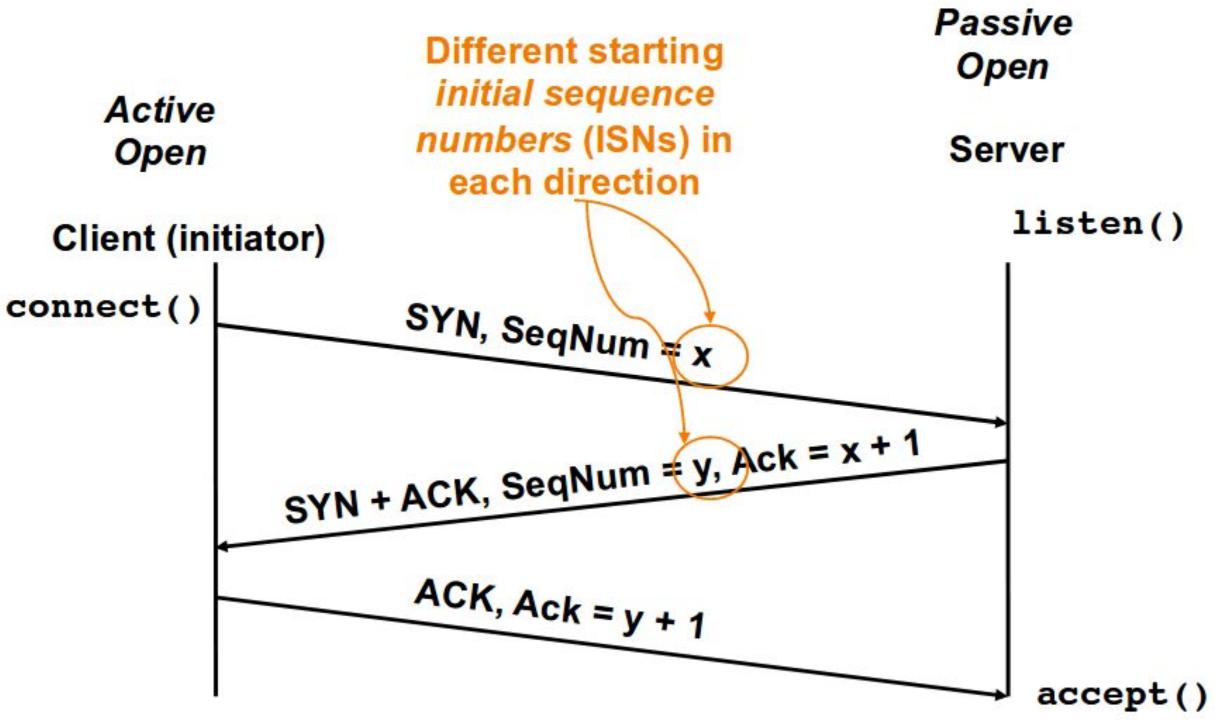


Each host tells its *Initial Sequence Number* (ISN) to the other host.

(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

Timing Diagram: 3-Way Handshaking

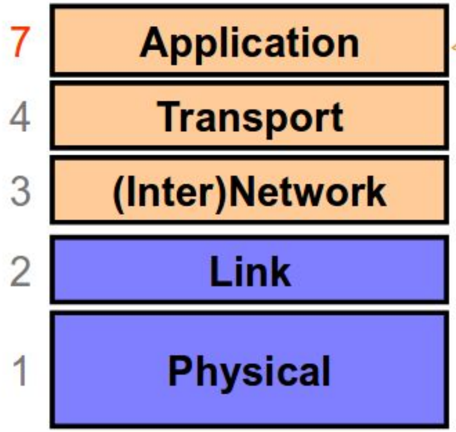


Active Open
Client (initiator)
connect()
Passive Open
Server
listen()
Different starting *initial sequence numbers* (ISNs) in each direction
SYN, SeqNum = x
SYN + ACK, SeqNum = y, Ack = x + 1
ACK, Ack = y + 1
accept()

TCP Issues

- TCP is not inherently secure
- 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
  - An attacker can also inject RST packets. If the connection information is correct, the receiving party will terminate the connection immediately.
  - An attacker can even create an entirely fake connection if they can see data being transmitted (on-path) or guess the connection information

Layer 7: Application Layer



Communication of whatever you wish

Can use whatever transport(s) is convenient

Freely structured

E.g.:

Skype, SMTP (email),
HTTP (Web), Halo, BitTorrent

Attacker Definitions (for the purposes of this class)

Man in the middle: Attacker can see and modify traffic (this includes dropping packets).

On path: Attacker can see but not modify traffic (Different definition than Dave Wagner’s semesters).

Off path: Attacker can’t see ****.

Remember all attackers can spoof whatever packets they want!

Think about what fields an attacker would need to guess correctly. (Hint, answer varies depending on the situation) [See also: Kaminksy attack]

Intrusion Detection

HIDS: Host based Intrusion Detection System

Benefits: Can read reconstructed data.

Drawbacks: Have to add to every host.

NIDS: Network based Intrusion Detection System

Benefits: Easy to add to a network (cheap and easy), don’t need to touch end systems.

Drawbacks: Can be evaded as an L3 device. (example, can’t decrypt https traffic)

Things to Review

DNS

Firewalls

Well I have questions, HAH

Spring 2017 - MT2

(q) When establishing a TCP connection, the client and the server engage in a three-way handshake to determine the shared Initial Sequence Number they will both use for that connection.

○ **True** ○ **False**

Spring 2017 - MT2

(q) When establishing a TCP connection, the client and the server engage in a three-way handshake to determine the shared Initial Sequence Number they will both use for that connection.

True [✓ **False**]

Solution: In TCP, the client and the server each select their own ISN; they do not share an ISN.

Problem 3. [Intrusion Response] (6 points)

The software company Snoracle (slogan: “Unwakeable”) is selling a new defense against DDoS attacks. Their software looks at the source IP address on all incoming packets, and if it finds any IP address that accounts for more than 1% of traffic over the last hour, it installs an entry in the router that blocks all packets from that address for the next 24 hours. Their marketing folks are claiming that this will stop all DDoS attacks cold in the water. Is this a good solution to the problem? Give one reason why or why not.

Problem 3. [Intrusion Response] (6 points)

The software company Snoracle (slogan: "Unwakeable") is selling a new defense against DDoS attacks. Their software looks at the source IP address on all incoming packets, and if it finds any IP address that accounts for more than 1% of traffic over the last hour, it installs an entry in the router that blocks all packets from that address for the next 24 hours. Their marketing folks are claiming that this will stop all DDoS attacks cold in the water. Is this a good solution to the problem? Give one reason why or why not.

No. It's a poor solution.

- *It's too easy to break: with more than 100 zombies, you can flood the victim's network link without any zombie consuming more than 1% of traffic.*
- *It's too easy to evade detection with forged source addresses. You just use a different (forged) IP address on every packet.*
- *It doesn't protect against attacks that overwhelm resources at the end host (e.g., CPU, memory) without filling the network pipe.*
- *Attackers could exploit this to cause collateral damage to innocent third parties. If CNN is using this, an attacker could prevent Joe from being able to reach CNN by sending a large number of packets whose IP address has been forged to look like they came from Joe.*

Any one of these was an acceptable reason.

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

(l) A useful property of fiber optic cables is that the technology fundamentally eliminates the possibility of eavesdropping.

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

(n) 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.

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

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

○ **True** [✓ ● **False**]

(l) A useful property of fiber optic cables is that the technology fundamentally eliminates the possibility of eavesdropping.

○ **True** [✓ ● **False**]

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

[✓ ● **True**] ○ **False**

(n) 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.

○ **True** [✓ ● **False**]

Solution: DNS responses have expiration dates. It is the client's responsibility to flush expired mappings.

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

○ **True** [✓ ● **False**]

(a) In ONE SENTENCE, explain the purpose of a RST injection attack: that is, what goal does an attacker try to accomplish by launching such an attack?

(b) What information is needed for an attacker to carry out a successful RST injection attack?

(c) Explain under what circumstances an off-path attacker can conduct a successful RST injection attack. If it is impossible for them to do so, explain why they cannot.

(d) Suppose an attacker launches a RST injection attack against Alice. Are there situations in which Alice can detect that the attack has occurred? If **YES**, explain how she might do so. If **NO**, explain why it's not possible for her to do so.

Problem 4 ***RST injection*** **(60 points)**

This problem concerns RST injection attacks.

(a) In ONE SENTENCE, explain the purpose of a RST injection attack: that is, what goal does an attacker try to accomplish by launching such an attack?

> **Solution:** In a RST injection attack, the attacker aims to disrupt an existing TCP connection, causing it to prematurely terminate.

(b) What information is needed for an attacker to carry out a successful RST injection attack?

> **Solution:** The attacker needs to know the target connection's 4-tuple (source and destination addresses and ports), and the current sequence number that the target expects to see from the spoofed source.
>
> A common error was to indicate that the attacker needs to know *both* sequence numbers (used in each direction).

(c) Explain under what circumstances an off-path attacker can conduct a successful RST injection attack. If it is impossible for them to do so, explain why they cannot.

> **Solution:** An off-path attacker can conduct a successful attack if they can somehow gather or infer the information specified in the previous question.
>
> Alternatively, a solution that states that randomized ISNs make the attack infeasible is also acceptable.

Suppose an attacker launches a RST injection attack against Alice. Are there situations in which Alice can detect that the attack has occurred? If YES, explain how she might do so. If NO, explain why it’s not possible for her to do so.

Solution: The answer we had intended was: YES. When the attacker injects their spoofed traffic, they cannot prevent any traffic sent by Alice’s legitimate peer (Bob) that’s already in flight from also arriving. Thus, Alice can observe both the receipt of a RST purportedly from Bob (the attack), as well as additional traffic (with later sequence numbers) arriving from Bob. Such a pattern does not make sense for the benign situation that Bob’s own system sent the RST.

Problem 6 *Coffee Shop Worries* (54 points)

Alice and Bob just arrived at Brewed Awakening, the local coffee shop. Eve is already there, enjoying a cup of tea.

(a) (6 points) Alice wants to connect to Brewed Awakening's WiFi network. Under which protocols would her connections be safe from sniffing attacks by other coffee shop visitors, such as Eve? **Mark all that apply.**

- ○ WEP
- ○ WPA2 - Enterprise mode
- ○ WPA2 - Personal mode
- ○ None of these

(b) (24 points) Turns out that Brewed Awakening's network has no encryption. Alice warns Bob that its not safe to use this connection, but Bob disagrees. Bob connects to the WiFi, and tests that he has Internet connectivity by going to `https://kewlsocialnet.com`. It loads without issues. Bob says the Alice: "See, no problem! That access was totally safe!"

If Bob is correct and the access to `kewlsocialnet.com` was safe, explain why he is correct. If he is not correct, provide a network attack against Bob.

Answer:

Problem 6 ***Coffee Shop Worries*** **(54 points)**

Alice and Bob just arrived at Brewed Awakening, the local coffee shop. Eve is already there, enjoying a cup of tea.

(a) (6 points) Alice wants to connect to Brewed Awakening's WiFi network. Under which protocols would her connections be safe from sniffing attacks by other coffee shop visitors, such as Eve? **Mark all that apply.**

○ WEP

[✓ ● WPA2 - Enterprise mode]

○ WPA2 - Personal mode

○ None of these

Solution: Only "WPA2 - Enterprise mode" provides per-connection secret keys with the WiFi access point to secure each connection separately.

(b) (24 points) Turns out that Brewed Awakening's network has no encryption. Alice warns Bob that its not safe to use this connection, but Bob disagrees. Bob connects to the WiFi, and tests that he has Internet connectivity by going to `https://kewlsocialnet.com`. It loads without issues. Bob says the Alice: "See, no problem! That access was totally safe!"

If Bob is correct and the access to `kewlsocialnet.com` was safe, explain why he is correct. If he is not correct, provide a network attack against Bob.

Solution: Bob is correct.

Bob is visiting an HTTPS website, which uses TLS to provide an end-to-end secure channel. As Bob's browser did not encounter any certificate warnings, then unless there's been a CA breach or some other CA issue, the network connection has confidentiality, authentication, and integrity.

We allowed full credit for solutions that specified that Bob was incorrect and provided a valid approach for undermining his HTTPS connection to the site, including the threat of obtaining fraudulent certs from misbehaving CAs.

We allowed only partial credit for solutions that framed attacks that would work in the situation if TLS did not provide all of the strong security properties that it does. These solutions received more credit if they clearly stated that the attack is relevant for Bob's *subsequent* connections, rather than his test connection. These solutions received less credit if they were simply stating that because the WiFi network is unencrypted, an attacker could read Bob's private information, since use of HTTPS prevents that.

(c) (24 points) Now that he has tested his WiFi access, Bob then tells Alice: "I want to buy that last muffin at the counter. Let me check if I have enough money in my bank account." Eve hears this and panics! She wants the last muffin too but is waiting for her friend Mallory to bring enough cash to buy it. She is now determined to somehow stop Bob from buying that last muffin by preventing him from checking his bank account. Through the corner of her eye, Eve sees Bob start to type `https://bank.com` in his browser URL bar ...

Describe two network attacks Eve can do to prevent Bob from checking his bank account. For each attack, describe clearly in one or two sentences how Eve performs the attack.

Attack #1:

Attack #2:

Solution:

Note that Eve cannot do an ARP or DHCP spoofing attack as Bob has already connected to the WiFi network, so already knows the IP and hardware addresses of the local network's gateway and DNS resolver. (This assumes that extraneous ARPs are not accepted by Bob's system. ARP spoofing is a viable answer for this problem if accompanied by specific mention of this consideration.)

1. TCP RST injection attack — Eve can sniff Bob's transmitted (and received) packets, so she can observe the sequence numbers of TCP packets. Thus, Eve can send a valid TCP RST packet to Bob's browser (or to the bank website), resetting the TCP connection.

2. DNS response spoofing — When Bob tries to load the bank website, his browser will generate a DNS request for the bank's domain. Eve can spoof a response with an incorrect answer, preventing Bob from loading the bank website properly.

3. DoS attack on either Bob's system or the coffee shop network. This can be done through various means, such as DNS amplification attacks directed at Bob.

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.

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.

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, both select their own ISN)

Hosts that use DHCP on a wired networking technology such as Ethernet are protected against possible DHCP spoofing attacks. (False, DHCP is broadcast.)

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).

Professor Raluca gets home after a tiring day writing papers and singing karaoke :). She opens up her laptop and would like to submit them to a conference. From a networking and web perspective, what are the steps involved in submitting her paper?

Raluca's computer needs to connect to the wifi. What messages are exchanged in the 4 part handshake in order to achieve this?

Raluca's computer sends: ___________. This message is broadcasted/unicasted (Choose one and explain):

A DHCP server replies with a DHCP Offer. What does this message contain? What can a malicious attacker do at this step? Keep in mind that an attacker on the same subnet can hear the discovery message.

Raluca's computer sends: _____________. This message is broadcasted/unicasted (Choose one and explain)

The server then responds with: ____________.

Raluca would like to print out her paper. Her printer is on a different local network with the IP address 192.168.1.5 and the MAC address: 1E:AT:DE:AD:BE:EF.

Raluca's computer is configured as follows:
IP Address: 192.168.0.2
DNS Server: 8.8.8.8
Subnet mask: 255.255.255.0
Default Gateway: 192.168.0.1
MAC Address: F8:DB:88:F8:4C:27

What address does Raluca's computer make an ARP request for? ____________

The response she gets back is: 16:1D:EA:DB:EE:F1.
Fill out the information for Raluca's packet below:

Raluca's Packet
Source IP address:
Destination IP:
Source MAC Address:
Destination MAC Address:

The router (router A) routes this packet to the router (router B) of the printer using the destination IP address. The MAC address for router B is C0:FF:EE:C0:FF:EE.
What address does the router B make an ARP request for? ___________

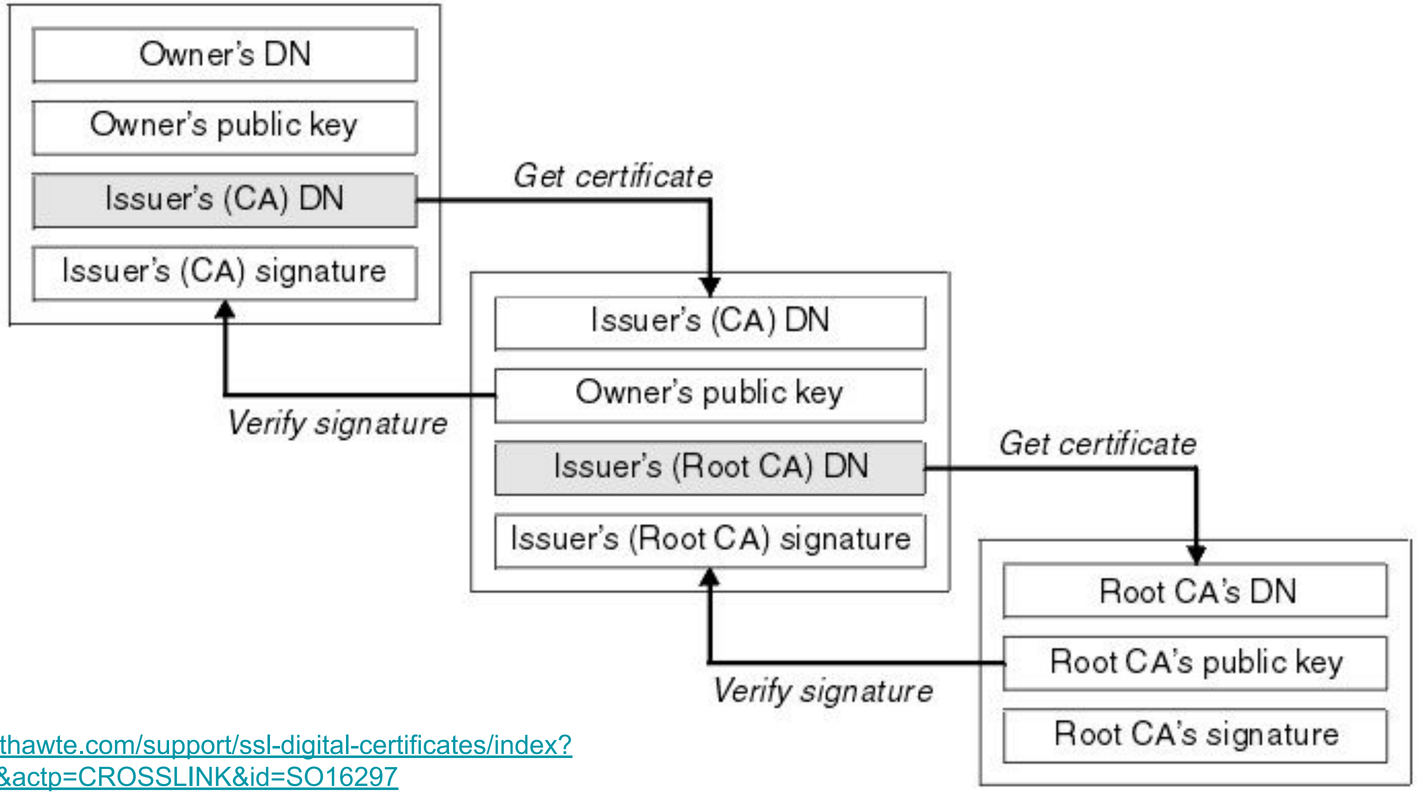
Oh no! Raluca has a smart refrigerator that has been taken over by an attacker \frownie{}. Assume her refrigerator is on her local network. How can the attacker intercept Raluca's paper before it gets to the printer?

Your answer here!

SSL/TLS

- Secure end to end secure communications channel (CIA)
    - Secure no matter what other agents in between do
- Browser must know the public key of the server
    - Certification used to distribute/verify public keys

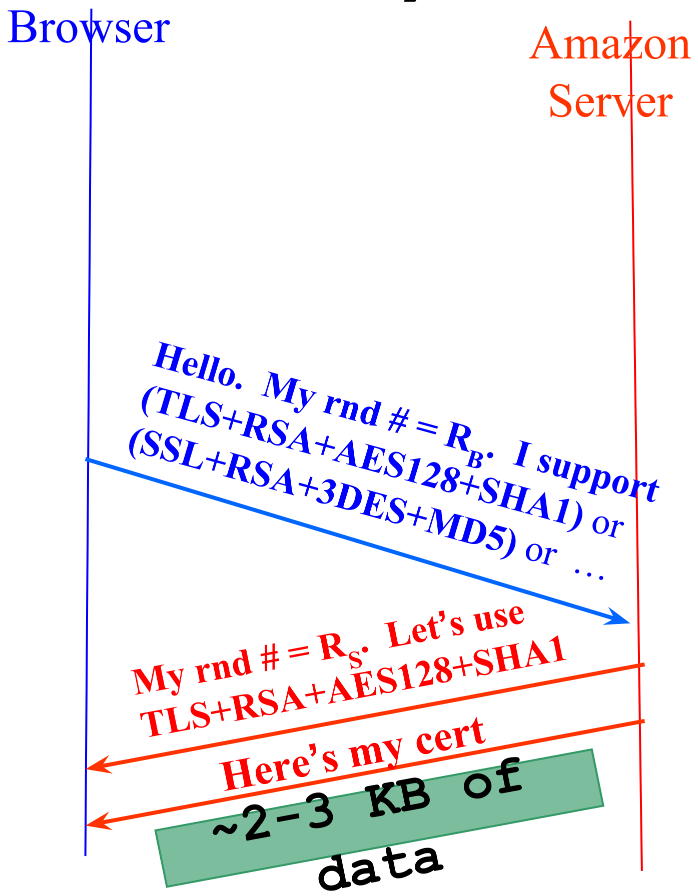
Certification / Certificate Chain



Source:
https://search.thawte.com/support/ssl-digital-certificates/index?page=content&actp=CROSSLINK&id=SO16297

HTTPS Connection (SSL / TLS)

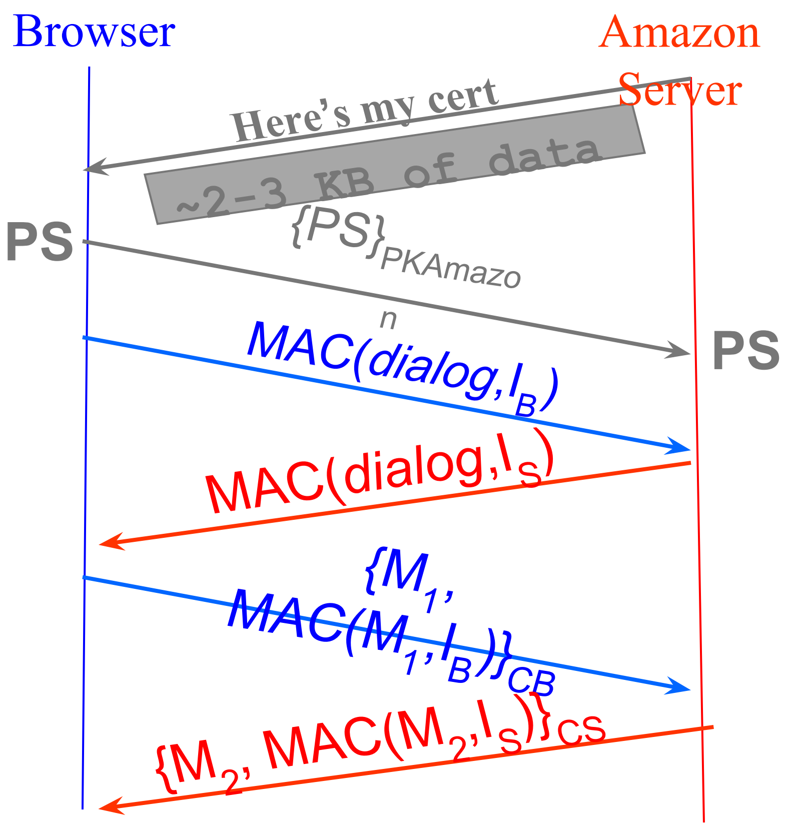
- Browser (client) connects via TCP to Amazon's **HTTPS** server
- Client picks 256-bit random number R_B, sends over list of crypto protocols it supports
- Server picks 256-bit random number R_S, selects protocols to use for this session
- Server sends over its certificate
- (all of this is in the clear)
- ***Client now validates cert***


Browser
Amazon Server
Hello. My rnd # = R_B. I support (TLS+RSA+AES128+SHA1) or (SSL+RSA+3DES+MD5) or …
My rnd # = R_S. Let's use TLS+RSA+AES128+SHA1
Here's my cert
~2-3 KB of data

Exchange with RSA

Q: Forward secrecy?
A: No forward secrecy because attacker can decrypt PS and knows R_B, and R_S and computes secrets

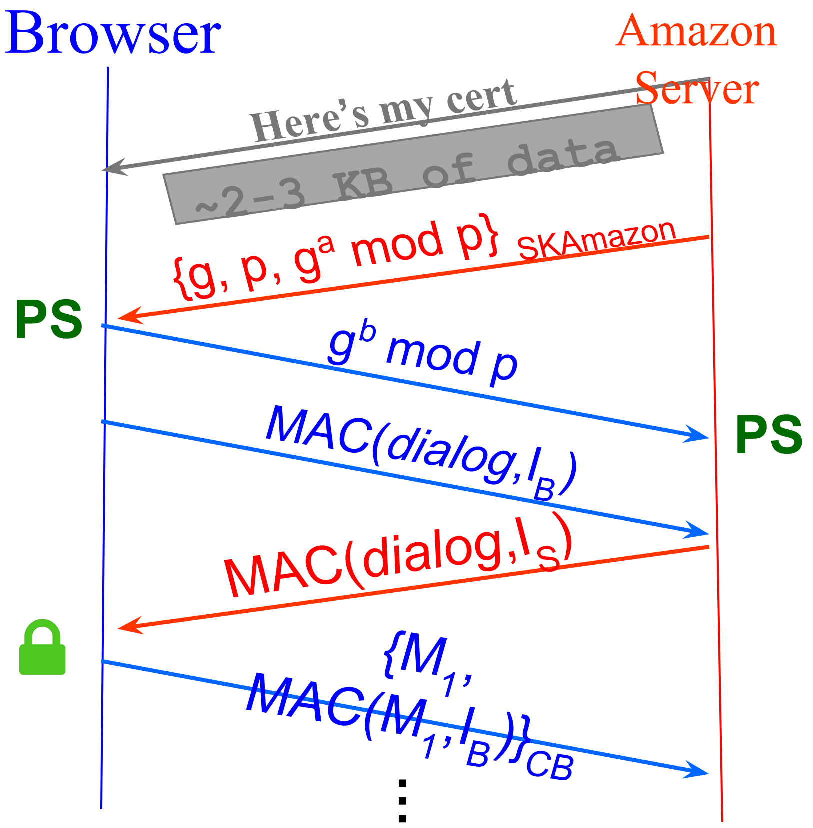
- For RSA, browser constructs "Premaster Secret" **PS**
- Browser sends PS encrypted using Amazon's public RSA key K_{Amazon}
- Using PS, R_B, and R_S, browser & server derive symm. *cipher keys* (C_B, C_S) & MAC *integrity keys* (I_B, I_S)
  - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far
- If good MAC, Browser displays
- All subsequent communication encrypted w/ 🔒 symmetric cipher (e.g., AES128) cipher keys in some chaining mode, MACs
  - Sequence #'s thwart replay attacks



Exchange via Diffie-Hellman

Q: Forward secrecy?
A: Has forward secrecy because shared secret never sent over the network! If attacker as SK_{Amazon}, cannot decrypt a.

- For Diffie-Hellman, server generates random a, sends public params and g^a mod p
  - Signed with server's private key
- Browser verifies signature using PK from certificate
- Browser generates random b, computes **PS** = g^{ab} mod p, sends to server
- Server also computes **PS** = g^{ab} mod p
- Remainder is as before: from PS, R_B, and R_S, browser & server derive symm. *cipher keys* (C_B, C_S) and MAC *integrity keys* (I_B, I_S), etc…

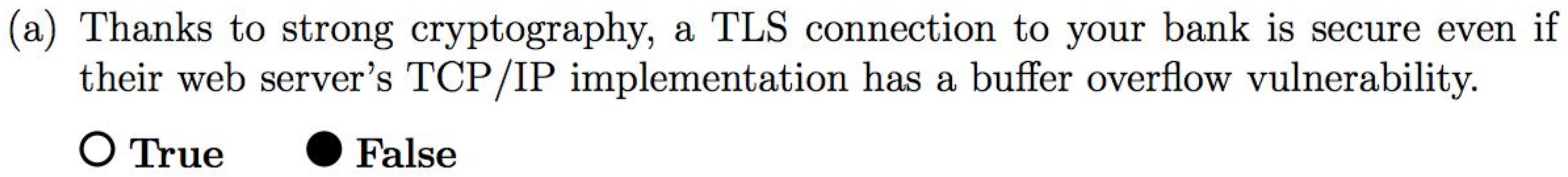


Spring 2017 - Final

(a) Thanks to strong cryptography, a TLS connection to your bank is secure even if their web server's TCP/IP implementation has a buffer overflow vulnerability.

○ **True** ○ **False**

Spring 2017 - Final



(a) Thanks to strong cryptography, a TLS connection to your bank is secure even if their web server's TCP/IP implementation has a buffer overflow vulnerability.

True [✓ False]

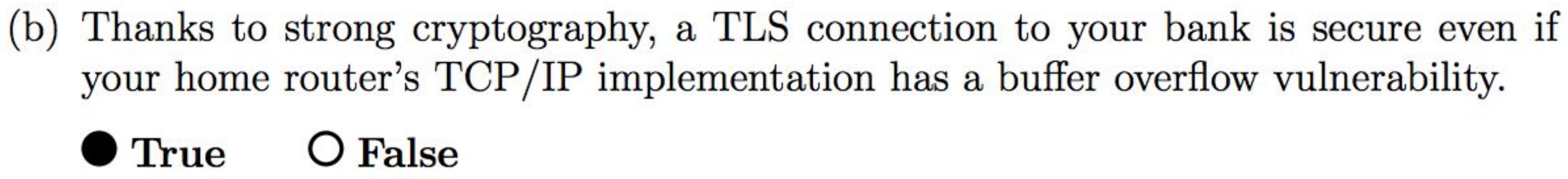
- TLS protects the channel, not the hosts. A buffer overflow vulnerability in the network stack could allow a malicious party to modify or exfiltrate traffic.

Spring 2017 - Final

(b) Thanks to strong cryptography, a TLS connection to your bank is secure even if your home router's TCP/IP implementation has a buffer overflow vulnerability.

○ **True** ○ **False**

Spring 2017 - Final

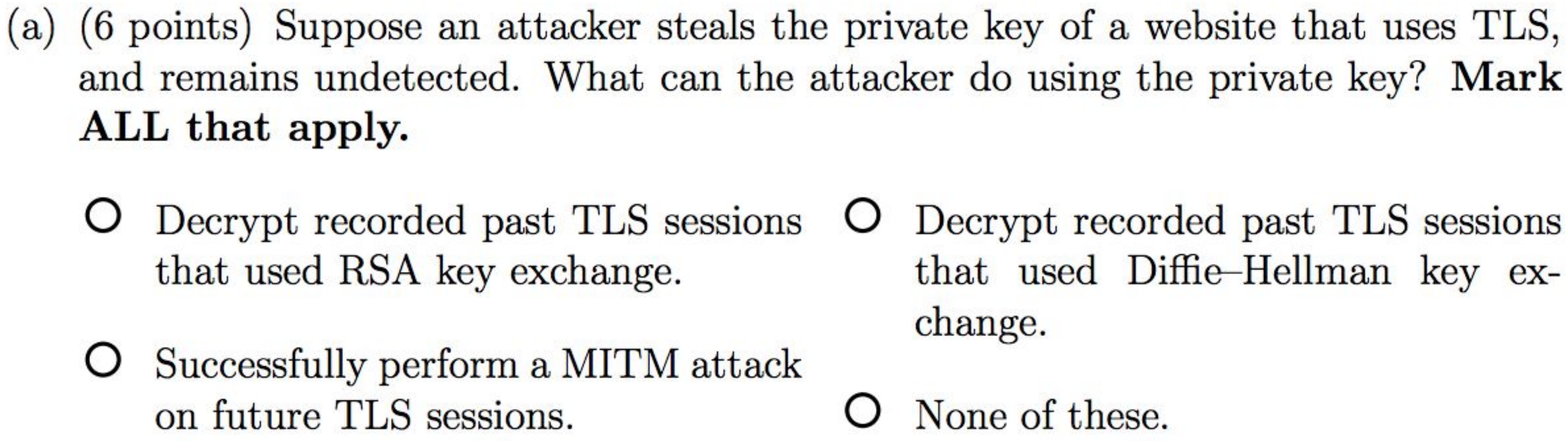


(b) Thanks to strong cryptography, a TLS connection to your bank is secure even if your home router's TCP/IP implementation has a buffer overflow vulnerability.

[✓ **True**] **False**

- A key property of TLS is how it provides end-to-end security: two systems can communicate using TLS without having to trust any of the intermediaries that forward their traffic. Thus, even if an attacker completely pwns your home router, the worst they can do to you is deny you service to your bank.

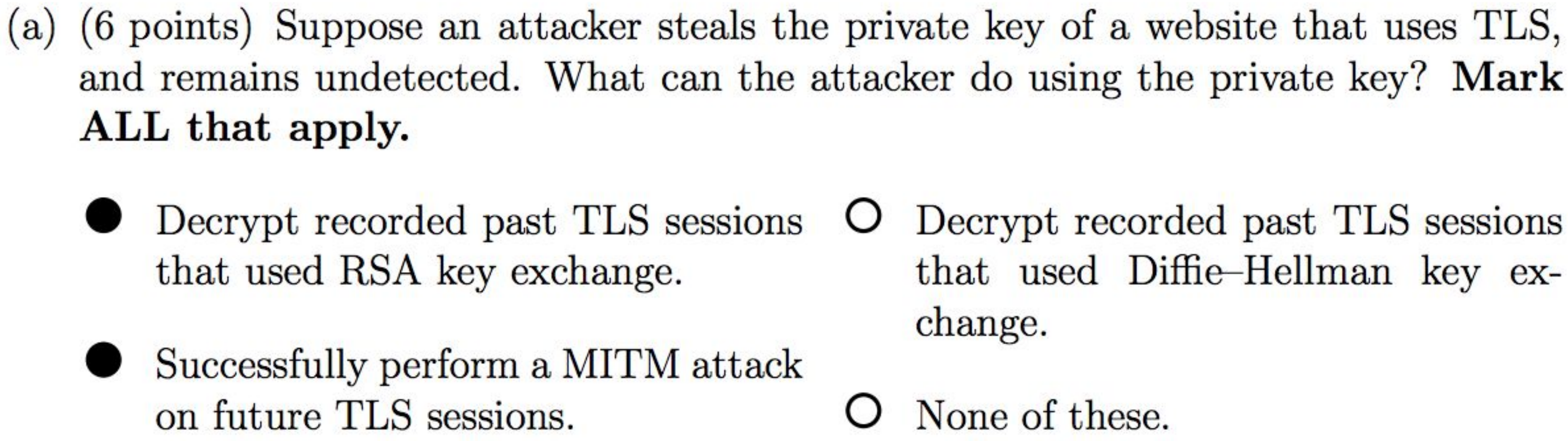
Spring 2017 - Final



(a) (6 points) Suppose an attacker steals the private key of a website that uses TLS, and remains undetected. What can the attacker do using the private key? **Mark ALL that apply.**

- ○ Decrypt recorded past TLS sessions that used RSA key exchange.
- ○ Successfully perform a MITM attack on future TLS sessions.
- ○ Decrypt recorded past TLS sessions that used Diffie–Hellman key exchange.
- ○ None of these.

Spring 2017 - Final



(a) (6 points) Suppose an attacker steals the private key of a website that uses TLS, and remains undetected. What can the attacker do using the private key? **Mark ALL that apply.**

- [✓ Decrypt recorded past TLS sessions that used RSA key exchange.]
- [✓ Successfully perform a MITM attack on future TLS sessions.]
- ○ Decrypt recorded past TLS sessions that used Diffie–Hellman key exchange.
- ○ None of these.

- RSA key exchange offers no forward secrecy, so all past sessions can be decrypted
- With the private key, a MITM can forge the server's signature. The MITM can negotiate a separate TLS connection to client and server, masquerading as the server to the client and vice versa

TLS Limitations/Issues

- The system requires us to trust Certificate Authorities
    a. Some of them are less than trustworthy
- Certificate management is complicated
    a. Expiring and replacing old certificates regularly
    b. Revoking leaked/compromised certificates quickly
    c. How does a browser know your website supports HTTPS?
- TLS can’t protect against logical errors on the client/server side like:
    a. Command injection vulnerabilities
    b. XSS vulnerabilities
    c. Other logical flaws
    d. TLS protects the CHANNEL, not the HOSTS

Web Security

“I asked my grad students for a joke about web security, and their response was: Isn’t web security already a joke?”

~Professor Raluca sp16, fa16, sp18

“You see, that was funny!”

~Professor Raluca sp18

What’s important here?

- Same origin policy
- COOOOKIESSSSSS
- Attacks! (XSS, injection, CSRF)

HTML

A language to create structured documents
One can embed images, objects, or create interactive forms

index.html

```
<html>
    <body>
        <div>
            foo
            <a href="http://google.com">Go to Google!</a>
        </div>
        <form>
            <input type="text” />
            <input type=”radio” />
            <input type=”checkbox” />
        </form>
    </body>
</html>
```

CSS (Cascading Style Sheets)

Style sheet language used for describing the presentation of a document

index.css

```
p.serif {
font-family: "Times New Roman", Times, serif;
}
p.sansserif {
font-family: Arial, Helvetica, sans-serif;
}
```

Javascript

Programming language used to manipulate web pages. It is a high-level, untyped and interpreted language with support for objects.

Supported by all web browsers

```
<script>
function myFunction() {
document.getElementById("demo").innerHTML = ”Text changed.";
}
</script>
```

Very powerful!

What can you do with Javascript?

Change HTML content, images, style of elements, hide elements, unhide elements, change cursor, read and change cookies.

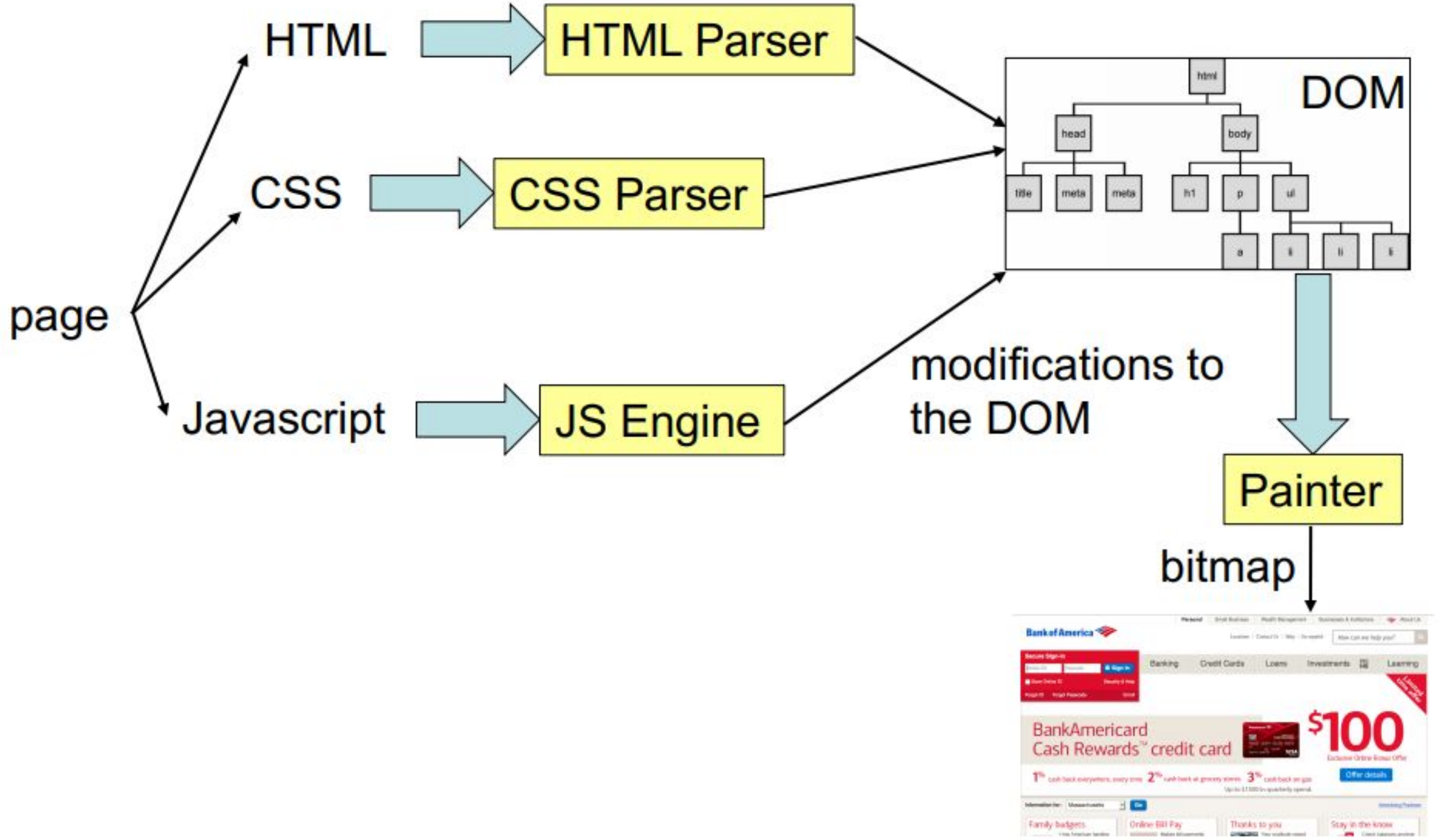
Read cookie with JS:

```
var x = document.cookie;
```

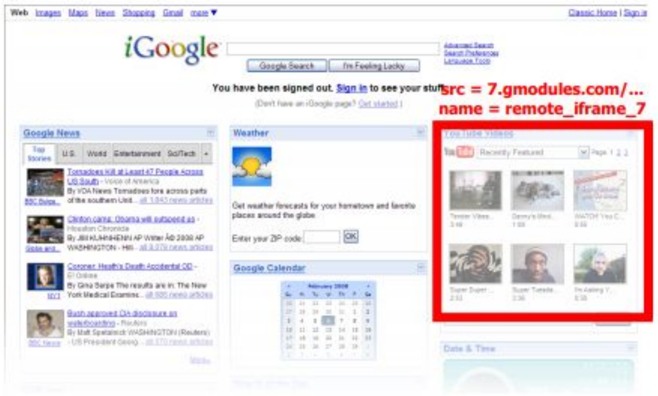
Change cookie with JS:

```
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
```

Page rendering



Frames



- Modularity
  - Brings together content from multiple sources
  - Client-side aggregation
- Delegation
  - Frame can draw only on its own rectangle

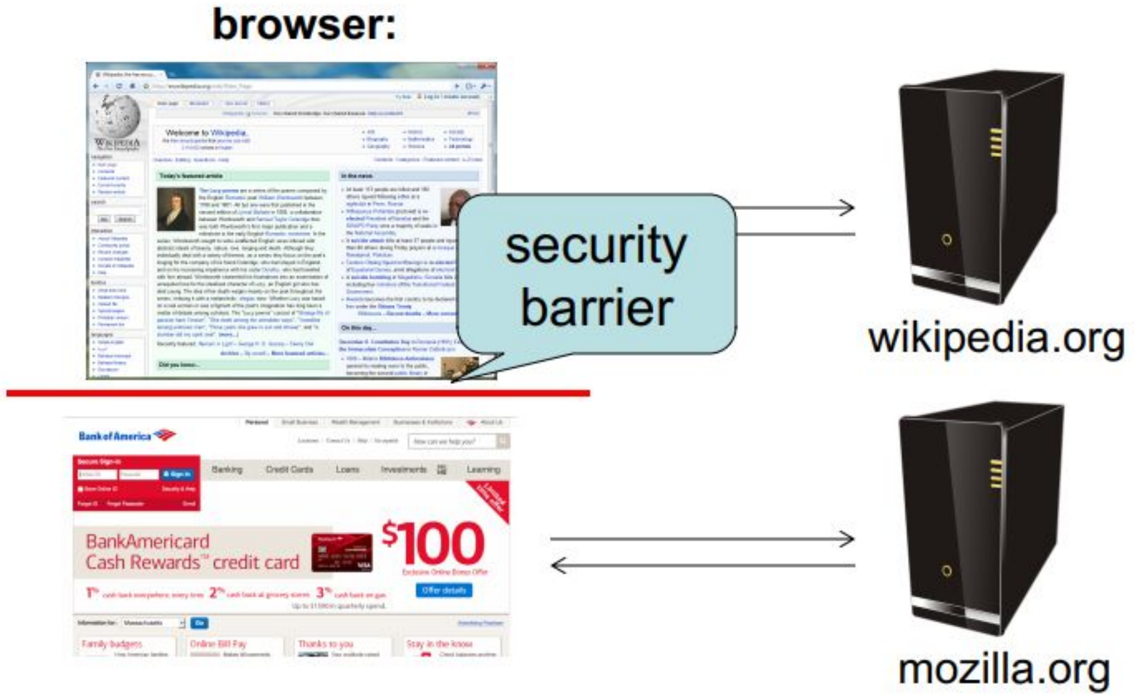
Outer page can specify only sizing and placement of the frame in the outer page.

Frame isolation: Our page cannot change contents of inner page. Inner page cannot change contents of outer page.

Slide from Dan Boneh

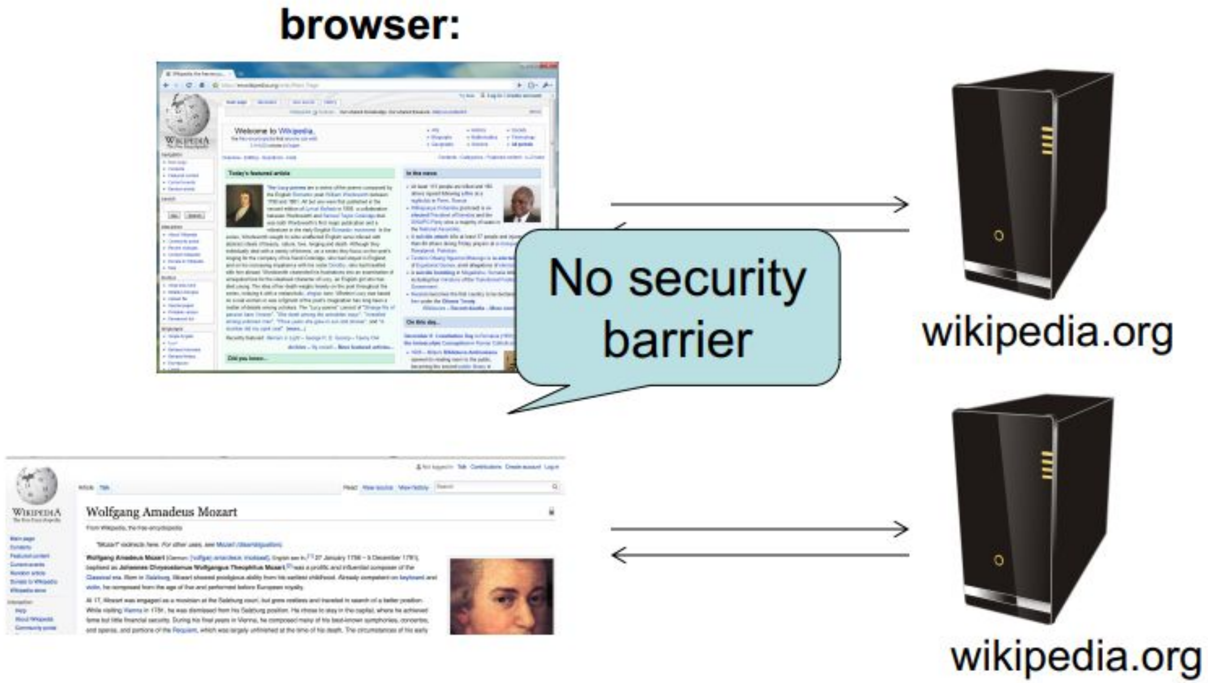
Same-origin policy

- Each site in the browser is isolated from all others



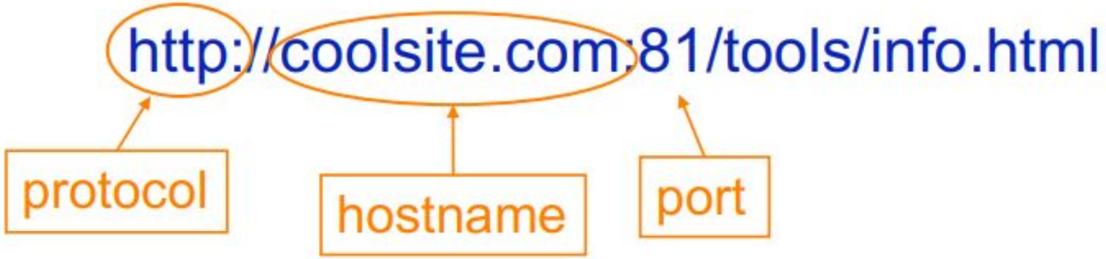
Same-origin policy

- Multiple pages from the same site are not isolated



Origin

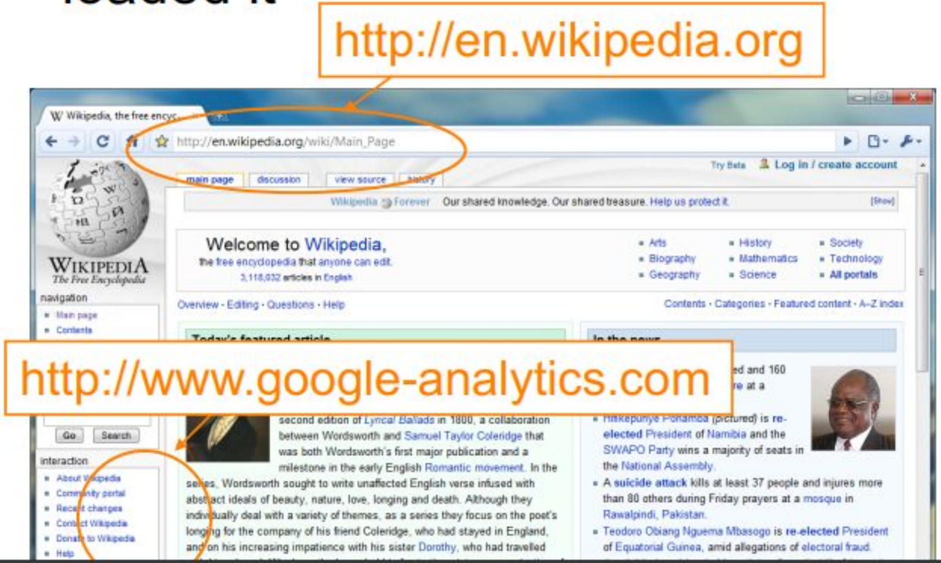
- Granularity of protection for same origin policy
- Origin = protocol + hostname + port



- It is **string matching**! If these match, it is same origin, else it is not. Even though in some cases, it is logically the same origin, if there is no match, it is not

Same-origin policy

- The origin of a page is derived from the URL it was loaded from
- Javascript runs with the origin of the page that loaded it


http://en.wikipedia.org
http://en.wikipedia.org/wiki/Main_Page
http://www.google-analytics.com

Origins of other components

- **<img src=“”>** the image is “copied” from the remote server into the new page so it has the origin of the embedding page (like JS) and not of the remote origin
- iframe: origin of the URL from which the iframe is served, and not the loading website.

Exercises

Originating document	**Accessed document**
http://wikipedia.org/**a**/	http://wikipedia.org/**b**/
http://wikipedia.org/	http://**www.**wikipedia.org/
http://wikipedia.org/	**https**://wikipedia.org/
http://wikipedia.org**:81**/	http://wikipedia.org**:82**/
http://wikipedia.org**:81**/	http://wikipedia.org/

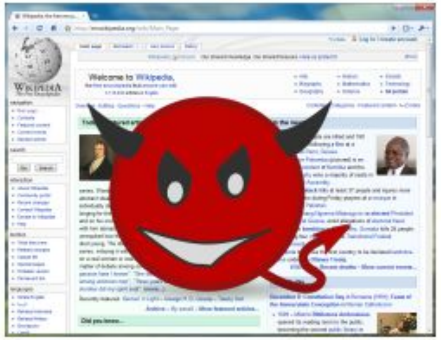
Exercises

Originating document	**Accessed document**	
http://wikipedia.org/**a**/	http://wikipedia.org/**b**/	✓
http://wikipedia.org/	http://**www.**wikipedia.org/	✗
http://wikipedia.org/	**https**://wikipedia.org/	✗
http://wikipedia.org**:81**/	http://wikipedia.org**:82**/	✗
http://wikipedia.org**:81**/	http://wikipedia.org/	✗

except !!!

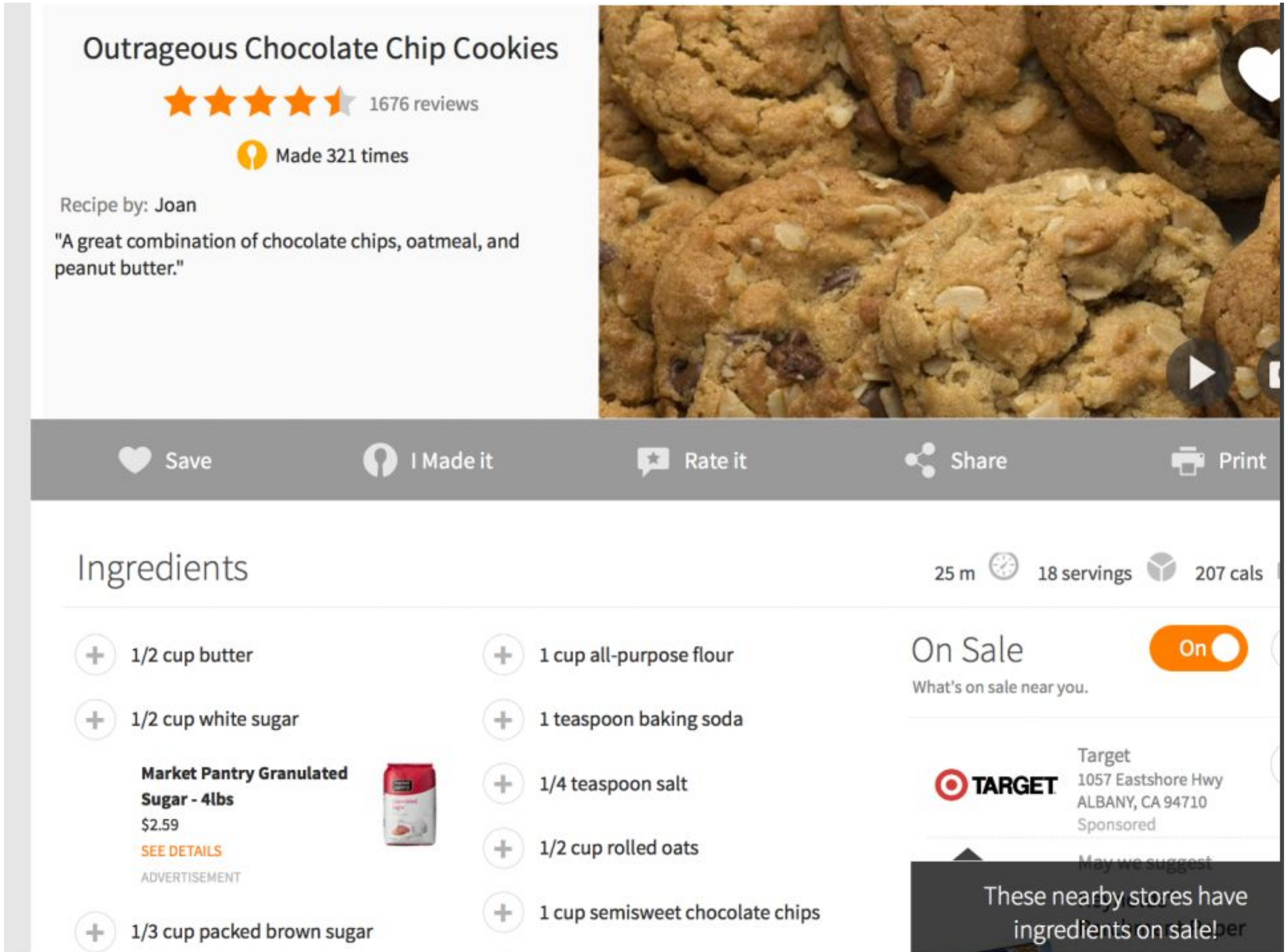
Cross-origin communication

- Allowed through a narrow API: **postMessage**
- Receiving origin decides if to accept the message based on origin (whose correctness is enforced by browser)



```
postMessage
("run this
script",
script)
```

Check origin, and request!



Outrageous Chocolate Chip Cookies

1676 reviews

Made 321 times

Recipe by: Joan

"A great combination of chocolate chips, oatmeal, and peanut butter."

Save | I Made it | Rate it | Share | Print

Ingredients

25 m | 18 servings | 207 cals

- 1/2 cup butter
- 1/2 cup white sugar

  Market Pantry Granulated Sugar - 4lbs
  $2.59
  SEE DETAILS
  ADVERTISEMENT

- 1/3 cup packed brown sugar
- 1 cup all-purpose flour
- 1 teaspoon baking soda
- 1/4 teaspoon salt
- 1/2 cup rolled oats
- 1 cup semisweet chocolate chips

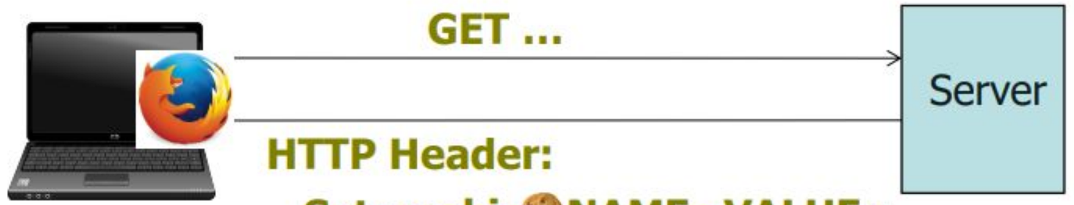
On Sale
On
What's on sale near you.

TARGET
Target
1057 Eastshore Hwy
ALBANY, CA 94710
Sponsored

May we suggest

These nearby stores have ingredients on sale!

Cookie scope



HTTP Header:

```
Set-cookie NAME=VALUE ;
    domain = (when to send) ;   scope
    path = (when to send)
    secure = (only send over SSL);
    expires = (when expires) ;
    HttpOnly
```

- **Expires is expiration date**
  - **Delete cookie by setting "expires" to date in past**
- **HttpOnly: cookie cannot be accessed by Javascript, but only sent by browser**

Cookie scope

- Scope of cookie might not be the same as the URL-host name of the web server setting it
  - **Different from same-origin policy!!**

Rules on:

1. What scopes a URL-host name is allowed to set
2. When a cookie is sent to a host

What scope a server may set for a cookie

The browser checks if the server may set the cookie, and if not, it will not accept the cookie.

domain: any domain-suffix of URL-hostname, except TLD

example: host = “login.site.com” [top-level domains, e.g. ‘.com’]

allowed domains	**disallowed domains**
login.site.com	**user.site.com**
.site.com	**othersite.com**
	.com

⇒ **login.site.com** can set cookies for all of **.site.com** but not for another site or TLD

path: can be set to anything

Examples

Web server at foo.example.com wants to set cookie with domain:

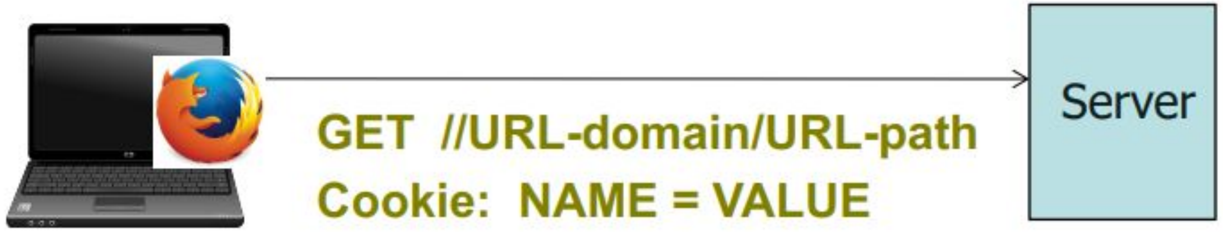
domain	Whether it will be set, and if so, where it will be sent to
(value omitted)	*foo.example.com* (exact)
bar.foo.example.com	
foo.example.com	**.foo.example.com*
baz.example.com	
example.com	
ample.com	
.com	

Examples

Web server at foo.example.com wants to set cookie with domain:

domain	Whether it will be set, and if so, where it will be sent to
(value omitted)	*foo.example.com* (exact)
bar.foo.example.com	Cookie not set: domain more specific than origin
foo.example.com	**.foo.example.com*
baz.example.com	Cookie not set: domain mismatch
example.com	**.example.com*
ample.com	Cookie not set: domain mismatch
.com	Cookie not set: domain too broad, security risk

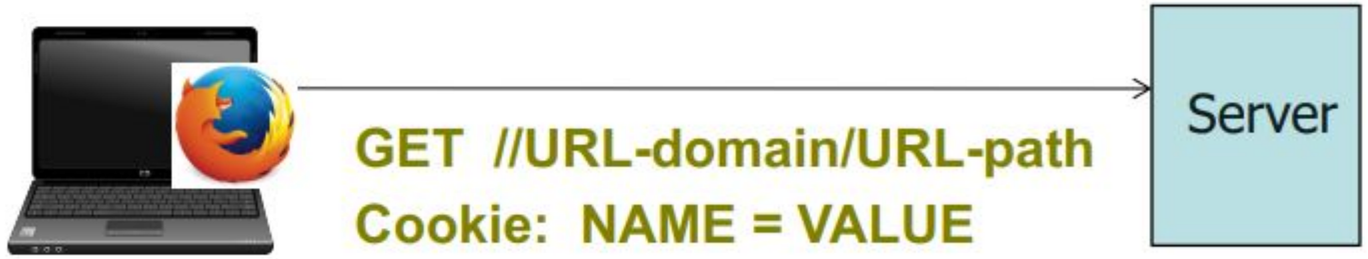
When browser sends cookie



Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain, and
- cookie-path is prefix of URL-path, and
- [protocol=HTTPS if cookie is "secure"]

When browser sends cookie



GET //URL-domain/URL-path

Cookie: NAME = VALUE

A cookie with

domain = example.com, and

path = /some/path/

will be included on a request to

http://foo.example.com/some/path/subdirectory/hello.txt

Examples: Which cookie will be sent?

cookie 1
name = userid
value = u1
domain = login.site.com
path = /
non-secure

cookie 2
name = userid
value = u2
domain = .site.com
path = /
non-secure

http://checkout.site.com/

http://login.site.com/

http://othersite.com/

Examples: Which cookie will be sent?

cookie 1	**cookie 2**
name = userid	**name = userid**
value = u1	**value = u2**
domain = login.site.com	**domain = .site.com**
path = /	**path = /**
non-secure	**non-secure**

http://checkout.site.com/	**cookie: userid=u2**
http://login.site.com/	**cookie: userid=u1, userid=u2**
http://othersite.com/	**cookie: none**

Examples

cookie 1
name = userid
value = u1
domain = login.site.com
path = /
secure

cookie 2
name = userid
value = u2
domain = .site.com
path = /
non-secure

http://checkout.site.com/	**cookie: userid=u2**
http://login.site.com/	**cookie: userid=u2**
https://login.site.com/	**cookie: userid=u1; userid=u2**
	(arbitrary order)

Cross-site scripting attack (XSS)

- Attacker injects a malicious script into the webpage viewed by a victim user
  - Script runs in user’s browser with access to page’s data

- The same-origin policy does not prevent XSS

Two main types of XSS

- *Stored* XSS: attacker leaves Javascript lying around on benign web service for victim to load
- *Reflected* XSS: attacker gets user to click on specially-crafted URL with script in it, web service reflects it back

XSS subverts the same origin policy

- Attack happens **within the same origin**
- Attacker tricks a server (e.g., `bank.com`) to send malicious script ot users
- User visits to `bank.com`

Malicious script has origin of bank.com so it is permitted to access the resources on bank.com

Cross Site Request Forgery (CSRF)

- Example:
  - User logs in to bank.com
    - Session cookie remains in browser state
  - User visits malicious site containing:

```
<form name=F action=http://bank.com/BillPay.php>
<input name=recipient value=badguy> …
<script> document.F.submit(); </script>
```

  - Browser sends user auth cookie with request
    - Transaction will be fulfilled
- Problem:
  - cookie auth is insufficient when side effects occur

CSRF Defenses

- Secret Validation Token

```
<input type=hidden value=23a3af01b>
```

- Referer Validation

```
Referer: http://www.facebook.com/home.php
```

- Others (e.g., custom HTTP Header)

```
X-Requested-By: XMLHttpRequest
```

Warm up questions

1) Summarize same-origin policy.

2) What is the interface through which two different tabs with different origins can talk to each other, in a way permitted by same-origin policy isolation?

3) Does same-origin policy protect against an XSS attack? Why or why not?

4) Does setting the secure flag (https only) on a cookie protect against a CSRF attack? Why or why not?

Warm up answers

1) A policy enforced by the browser that isolates the resources of an origin from another, where an origin is defined by protocol+host+port.

2) postMessage allows sending messages between origins. The receiving origin needs to accept this message.

3) Same-origin policy does not protect against XSS because the attack is carried within the same origin.

4) Setting the secure flag does not protect against a CSRF attack because in this attack, the browser automatically attaches the cookie to the request (as long as the attacker used a https request).

Problem 2 *Fill in the blank* (8 points)

Fill in the blank.

(a) When you click on a link, the ____________________ header tells the server which URL you were at that took you to the link.

(b) An attack that reconstructs what you are typing on your keyboard by recording and then analyzing the specific sounds made as you type would be an example of a ______________________________ attack.

(c) To evaluate an intrusion detection system, we usually need to know its false _______________ rate, its false _______________ rate, the base rate of attacks, and the relative cost of a ______________________________ vs. a ______________________________.

Problem 2 ***Fill in the blank*** **(8 points)**

Fill in the blank.

(a) When you click on a link, the $\boxed{\text{Referer}}$ header tells the server which URL you were at that took you to the link.

(b) An attack that reconstructs what you are typing on your keyboard by recording and then analyzing the specific sounds made as you type would be an example of a $\boxed{\text{side channel}}$ attack.

(c) To evaluate an intrusion detection system, we usually need to know its false $\boxed{\text{positive}}$ rate, its false $\boxed{\text{negative}}$ rate, the base rate of attacks, and the relative cost of a $\boxed{\text{false positive}}$ vs. a $\boxed{\text{false negative}}$.

Problem 10 ***Web security*** **(24 points)**

FaceSpace has implemented an all-new friend finder feature! Now, users of FaceSpace can enter a search string such as "`rohin`" in order to find people to friend. FaceSpace also keeps track of the most popular search strings. Since the feature is new, the most popular search strings at the moment have only been searched for a few hundred times.

When the user visits a URL such as
`https://www.facespace.com/friendfinder.php?name=rohin`
(note the use of HTTPS), Facespace runs the following (in pseudocode):

```
name = getParameterFromUrl(url, 'name');
increment_number_of_searches(name);
command = "SELECT username FROM users WHERE name = '" + name + "'";
results = execSQLForTable(command, "users");
html = "<p>You searched for: " + name + ".</p><p>" + results + "</p>";
send_to_client(html);
```

The code above first extracts "`rohin`" from the URL (simply by scanning for "`name=`" and returning whatever is after that). It issues an SQL query to the `users` table, which contains the username, password (in cleartext), and name of all of the FaceSpace users. The developers have made sure to ensure that the SQL command can access only the "`users`" table. The code sends back the results to the user.

When a user asks for the most popular search strings, FaceSpace goes through the database containing the number of searches for each search string, and returns the top 100 search strings sorted by number of searches.

- Alice is a FaceSpace user. She does not care about the most searched names, so she will never view the top 100 search queries. However, since she knows Mallory, she will visit any site that Mallory sends to her.
- Bob is another FaceSpace user. He does not know Mallory and is cautious by nature, so he will not visit any sites recommended by Mallory. He likes to follow celebrities and so views the top 100 search queries page frequently.
- Charlie is another FaceSpace user. He will not visit any sites recommended by Mallory and does not care about the most searched names, so he will never view the top 100 search queries. Like the Governor of Project 3, he is very careful and if anything seems suspicious, he will close his browser and go do something else.
- Mallory is an off-path attacker who has an account on FaceSpace.

For each of the following goals, say whether or not Mallory can achieve that goal: yes or no. If your answer is "yes" (i.e., Mallory can achieve the goal), then list the name of the attack technique she could use. You don't need to describe the attack in detail; just the name. If your answer is "no", you don't need to provide any further justification or explanation.

Do not assume any software vulnerabilities or design flaws in anyone's software, other than what is implied by this question.

(a) Mallory wants to steal Alice's FaceSpace cookie.

Can she?
Attack name:

(b) Mallory wants to steal Bob's FaceSpace cookie.

Can she?
Attack name:

(c) Mallory wants to steal Charlie's FaceSpace cookie.

Can she?
Attack name:

(d) Mallory wants to steal Alice's FaceSpace password.

Can she?
Attack name:

(e) Mallory wants to steal Bob's FaceSpace password.

Can she?
Attack name:

(f) Mallory wants to steal Charlie's FaceSpace password.

Can she?
Attack name:

Finally, answer the following question:

(g) FaceSpace hires a security auditor, who recommends that they add an unpredictable CSRF token to the search form and search URL. Thus, the search URL now looks like

```
https://www.facespace.com/friendfinder.php?token=1A0B743FC08DA37A&name=rohin
```

The code for that page is modified to first check that the token is correct; if it is incorrect or missing, none of the rest of the code is executed and the page doesn't load. Nothing is changed about the code for the page with the top 100 search strings.

Which of goals (a)–(f) become impossible, once this change is made? Do not justify your answer; just list the set of goals that Mallory cannot achieve.

(a) Mallory wants to steal Alice's FaceSpace cookie.

Solution: Can she? Yes
Attack name: reflected XSS

Comment: The attack is to send Alice a link to a search results page where the name contains Javascript, e.g., `https://www.facespace.com/friendfinder.php?name=<SCRIPT>...</SCRIPT>`.

We'll also accept "stored XSS", since Mallory could perform the attack in part (b) and then send Alice a link to the top-100 page.

(b) Mallory wants to steal Bob's FaceSpace cookie.

Solution: Can she? Yes
Attack name: stored XSS

Comment: Mallory could perform a bunch of repeated searches for the same name to get it listed on the top-100 page—sneakily choosing a name that contains Javascript—and then wait for Bob to visit the top-100 page.

(c) Mallory wants to steal Charlie's FaceSpace cookie.

Solution: Can she? No

(d) Mallory wants to steal Alice's FaceSpace password.

Solution: Can she? Yes
Attack name: SQL injection

Comment: Mallory searches for a name like
`nosuchuser'; SELECT password FROM users WHERE name = 'Alice`
and the results page will reveal Alice's password.

Or, she could search for something like
`'; SELECT * FROM users; --`
and the results page might reveal everyone's password.

(e) Mallory wants to steal Bob's FaceSpace password.

Solution: Can she? Yes
Attack name: SQL injection

(f) Mallory wants to steal Charlie's FaceSpace password.

Solution: Can she? Yes
Attack name: SQL injection

Finally, answer the following question:

(g) FaceSpace hires a security auditor, who recommends that they add an unpredictable CSRF token to the search form and search URL. Thus, the search URL now looks like

`https://www.facespace.com/friendfinder.php?token=1A0B743FC08DA37A&name=rohin`

The code for that page is modified to first check that the token is correct; if it is incorrect or missing, none of the rest of the code is executed and the page doesn't load. Nothing is changed about the code for the page with the top 100 search strings.

Which of goals (a)–(f) become impossible, once this change is made? Do not justify your answer; just list the set of goals that Mallory cannot achieve.

Solution: None. Nothing changes.

Comment: Nothing changes for (b)–(f). For (a), the reflected XSS attack is no longer possible, but the stored XSS attack remains possible. However, we ultimately decided to generously give full credit for the answer "(a) becomes impossible", since the stored XSS attack is a bit obscure. Thus, we graded you only on your ability to recognize that this change does not affect any of goals (b)–(f), and we ignored what you said about (a).

Rest in Peace (DON’T WORRY IT’S NOT BROCCOLI)