COMP435: *SECURITY CONCEPTS!*

Lecture 21: Web Security Cont., Cross-Site Forgery Attacks

tinyurl.com/comp435-fa25

Quiz Topics!

- Network Security Topics! Especially…
  - *Firewalls &* packet filters – reading a packet filter table & interpreting the rules! Filling in the rules to meet a specification;
  - *Intrusion Detection & Intrusion Prevention Systems.* Given some initial params, should be able to calculate alarm precision, total no. of alarms, FPR, TPR, etc…
  - *The Network Stack!* The different levels of abstraction, what is the interface at each level, what does data look like & what does the payload look like at each
    - TCP 3-way handshake!
  - *Secure Tunnels*: How do they work conceptually. VPNs. How does the SSH protocol work.

HYPERTEXT TRANSFER PROTOCOL (HTTP)

HyperText Transfer Protocol (HTTP)

Def’n: Set of rules that dictate how browsers and servers exchange information

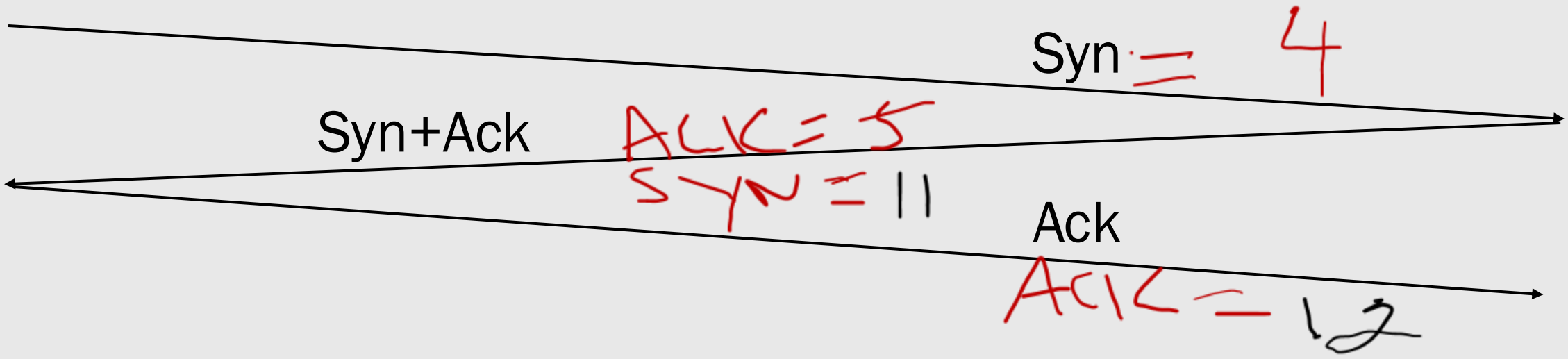
https://www.cs.unc.edu/~kakiryan/teaching.html

NSlookup www.cs.unc.edu

1. browser gets hostname from URL: www.cs.unc.edu
2. browser does DNS lookup to get IP address: 152.2.131.128
3. browser gets port number from URL: 80 default :)

https://www.cs.unc.edu/~kakiryan/teaching.html

4. browser establishes TCP connection to server



https://www.cs.unc.edu/~kakiryan/teaching.html

5. browser sends HTTP request to server

IP header (Dest addr: 152.19.160.128)
TCP header (Dest port: 80)
GET /~kakiryan/teaching/html HTTP/1.1 Referer: http://www.cs.unc.edu/~kakiryan/index.html User-Agent: Mozilla/5.0

https://www.cs.unc.edu/~kakiryan/teaching.html

5. browser sends HTTP request to server

IP header

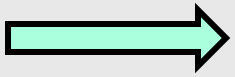
IP header (Dest addr: 152.19.160.128)
TCP header (Dest port: 80)
GET /~kakiryan/teaching/html HTTP/1.1 Referer: http://www.cs.unc.edu/~kakiryan/index.html User-Agent: Mozilla/5.0

IP packet

https://www.cs.unc.edu/~kakiryan/teaching.html

5. browser sends HTTP request to server

TCP header



IP header (Dest addr: 152.19.160.128)
TCP header (Dest port: 80)
GET /~kakiryan/teaching/html HTTP/1.1 Referer: http://www.cs.unc.edu/~kakiryan/index.html User-Agent: Mozilla/5.0

IP packet

https://www.cs.unc.edu/~kakiryan/teaching.html

5. browser sends HTTP request to server

application data

IP header (Dest addr: 152.19.160.128)
TCP header (Dest port: 80)
GET /~kakiryan/teaching/html HTTP/1.1 Referer: http://www.cs.unc.edu/~csturton/index.html User-Agent: Mozilla/5.0

IP packet

https://www.cs.unc.edu/~kakiryan/teaching.html

6. server responds

IP header (Src addr: 152.19.160.128, Dest addr...)

TCP header (Src port: 80, Dest port:...)

HTTP/1.1 200 OK

<html><body>....

https://www.cs.unc.edu/~kakiryan/teaching.html

7. browser renders HTML page:

Home Research Teaching Reading Group CV

Instructor of Record
- COMP 435: Computer Security Concepts (Fall 2025)
- COMP 311: Computer Organization (Fall 2025)
- COMP 520: Compilers (Summer Session II 2024)
- COMP 210: Data Structures and Analysis (Summer Session I 2024)
- COMP 210: Data Structures and Analysis (Summer 2022)
- COMP 110: Introduction to Programming and Data Science (Summer Session I 2021)

Graduate Teaching Assistant
- COMP 435: Computer Security Concepts (Fall 2023)
  - Guest Lecture on User Authentication
- COMP 435: Computer Security Concepts (Fall 2022)
  - Guest Lecture on Block Ciphers and Modes of Operation

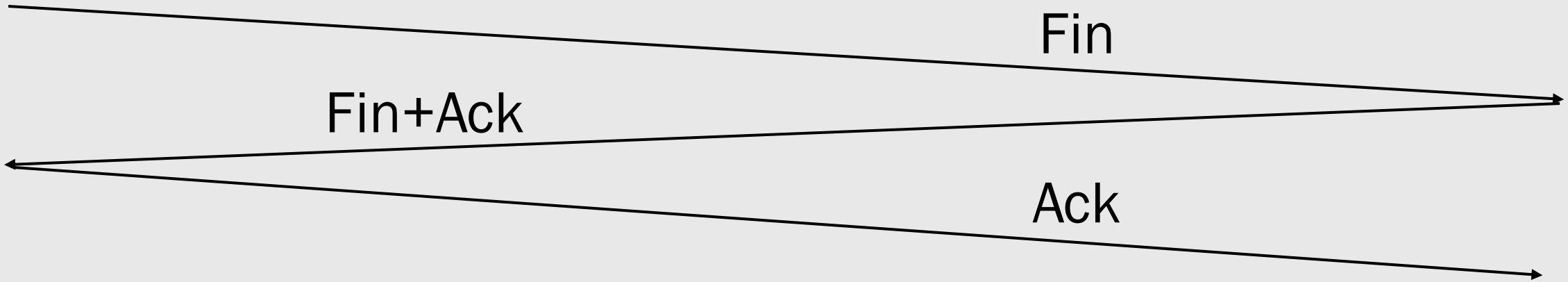
Head Teaching Assistant
- COMP 110: Introduction to Programming and Data Science (Fall 2021)
  - Guest Lecture on Recursion
- COMP 110: Introduction to Programming and Data Science (Spring 2021)
- COMP 110: Introduction to Programming and Data Science (Fall 2020)
- COMP 110: Introduction to Programming and Data Science (Summer Session II 2020)

Undergraduate Teaching Assistant
- COMP 211: Systems Fundamentals (Spring 2020)
- COMP 290: Tools for Computer Science (Spring 2020)
- COMP 110: Introduction to Programming and Data Science (Fall 2017-Fall 2019)
- COMP 101: Introduction to Computing (Spring 2018-Spring 2019)

Awards
- John M. Glotzer Graduate Teaching Assistant Award (2020-2021)

https://www.cs.unc.edu/~kakiryan/teaching.html

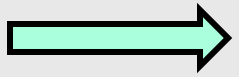
8. TCP connection is torn down



HTTP Request

```
GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0

optional-body
```

HTTP Request

request line 

```
GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0

optional-body
```

HTTP Request

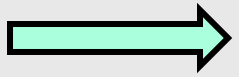
request line

other methods:
POST, CONNECT,
DELETE

```
GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0


optional-body
```

HTTP Request

request line



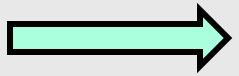
URL

- relative
- includes ?query

```
GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0

optional-body
```

HTTP Request

request line



GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0

optional-body

HTTP version

- 2.0 is newest
- 3.0 is in progress

HTTP Request

request headers ⟹

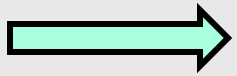
others:
Referer, Accept-Language, Expires

GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0

optional-body

HTTP Request

request body 

Not used with GET

```
GET /~kakiryan/teaching.html HTTP/1.1

Cookie: firstvisit=20210817
User-Agent: Mozilla/5.0

optional-body
```

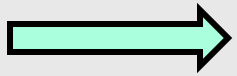
HTTP Response

```
HTTP/1.1 200 OK

Set-Cookie: lastvisit=20211001

<html><title>...
```

HTTP Response

status line

others:
404 Not Found



HTTP/1.1 200 OK

Set-Cookie: lastvisit=20211001

<html><title>...

HTTP Response

response header

others:
Location,
Allow

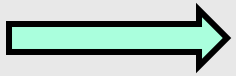
HTTP/1.1 200 OK

Set-Cookie: lastvisit=20211001

<html><title>...

HTTP Response

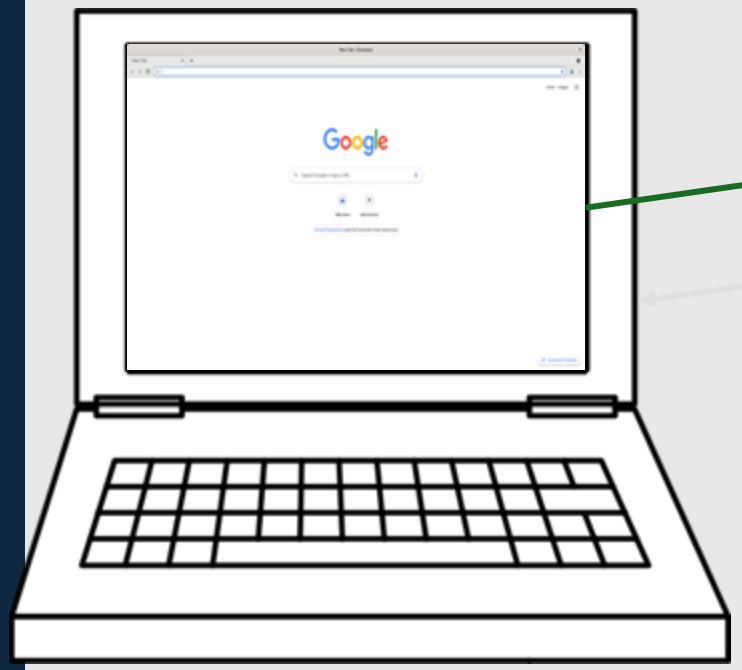
body
e.g., HTML page



HTTP/1.1 200 OK

Set-Cookie: lastvisit=20211001

<html><title>...



request

response

cs.unc.edu

mail.google.com

www.youtube.com

Ok, you've made a request, what gets returned in the response?!

HTML

Hypertext Markup Language (HTML)

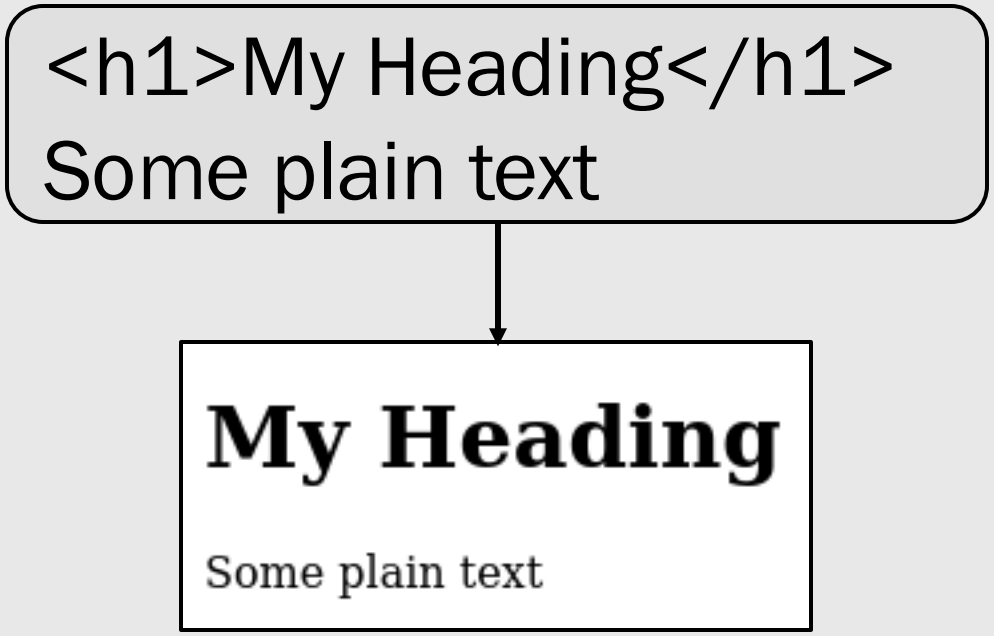
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html> Scroll
  <head>…</head>
  <body>
    <div class="center">
      <div class="nav">…</div>
      <!--end nav-->
      <div class="top-c">…</div>
      <div class="content"> = $0
        <ul>
          <b> Instructor of Record </b>
          <li>
            <a href="435-fa25/435-fa25.html">COMP 435: Computer Security Concepts (Fall 2025)</a>
          </li>
          <li>
            <a href="311-fa25/311-fa25.html">COMP 311: Computer Organization (Fall 2025)</a>
          </li>
          <li>
            <a href="archive/summer-520.html">COMP 520: Compilers (Summer Session II 2024)</a>
          </li>
          <li>
            <a href="archive/summer-210.html">COMP 210: Data Structures and Analysis (Summer Session I 2024)</a>
          </li>
          <li>
            <a href="archive/summer-x-210.html">COMP 210: Data Structures and Analysis (Summer 2022)</a>
          </li>
          <li>
            <a href="https://21ss1.comp110.com">COMP 110: Introduction to Programming and Data Science (Summer Session I 2021)
            </a>
          </li>
        </ul>
        <ul>…</ul>
        <ul>…</ul>
        <ul>…</ul>
        <ul>…</ul>
      </div>
    </div>
  </body>
</html>
```

Hypertext Markup Language (HTML)

- Tags in the text provide annotation
- Browser renders the annotated text

Hypertext Markup Language (HTML)

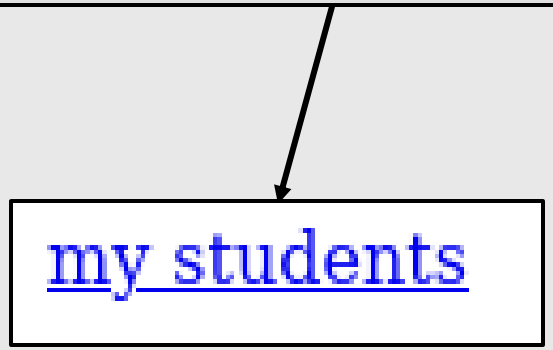
- Tags in the text provide annotation
- Browser renders the annotated text



Hypertext Markup Language (HTML)

- Tags in the text provide annotation
- Browser renders the annotated text

My Students



Hypertext Markup Language (HTML)

- Tags in the text provide annotation
- Browser renders the annotated text
- Executable content is allowed

```
<script>
executable code fragment goes here
</script>
```

Javascript

<script> block

javascript: scheme

event handler

Javascript

<script> block

javascript: scheme

event handler

<script>document.write(Date())</script>

<script src="*url*"></script>

Javascript

<script> block

javascript: scheme

event handler

Click Here

Click Here

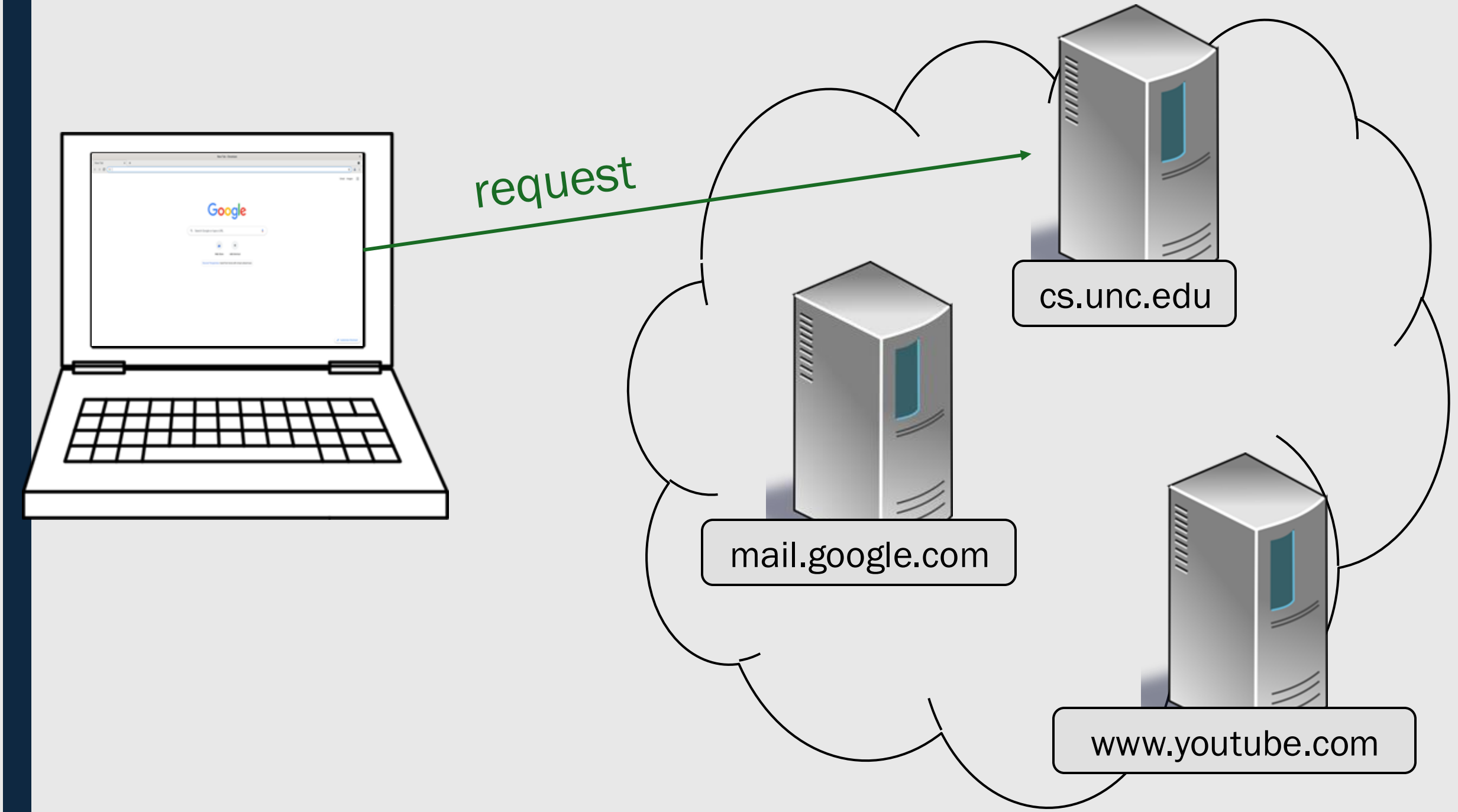
Javascript

<script> block

javascript: scheme

event handler

```
<button type="button" onclick="executable javascript">
button text</button>

<div onmouseover="executable javascript">some text</div>
```


request
cs.unc.edu
mail.google.com
www.youtube.com

BROWSER REDIRECTION

Browser Redirection

Def’n: a browser is forwarded to a new site and loads a page other than the one originally requested

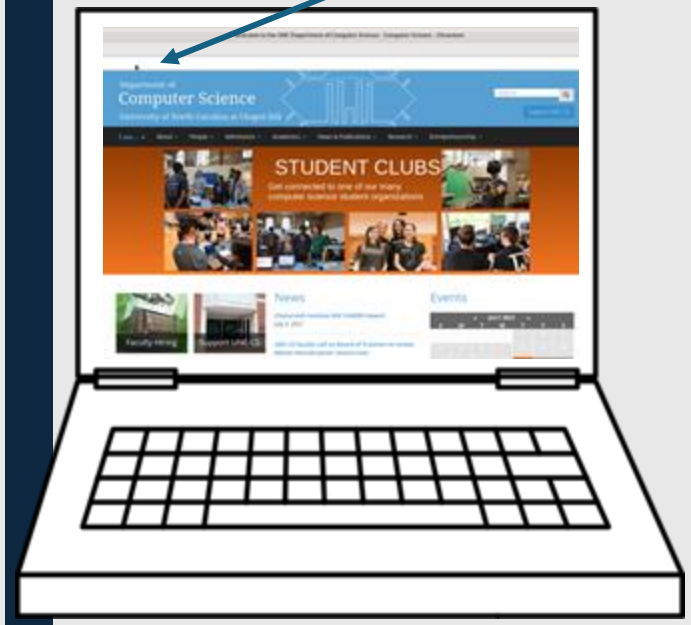
cs.unc.edu

request

response

cs.unc.edu

cs.unc.edu



cs.unc.edu

Browser Redirection

- JavaScript redirect
- Refresh meta tag
- Refresh header
- HTTP redirection

Browser Redirection

- JavaScript redirect
- Refresh meta tag
- Refresh header
- HTTP redirection

JavaScript Redirect in HTML

```
<script>window.location="url"</script>

<script>window.location.href="url"</script>
```

Refresh Meta Tag in HTML

```
<head>
<meta http-equiv="refresh" content="N; URL=new-url">
</head>
```

Browser Redirection

- JavaScript redirect
- Refresh meta tag
- Refresh header
- HTTP redirection

in HTTP response

Refresh Header in HTTP Response

HTTP/1.1 200 OK

Refresh: N; url=*new-url*

<html><title>...

HTTP Redirection in HTTP Response

HTTP/1.1 301 Moved Permanently

Location: *new-url*

CROSS SITE FORGERY REQUEST

Cross-Site Request Forgery (CSRF)

Def’n: An attacker injects a transaction request to a site from an authenticated user’s browser

a.k.a. XSRF, session riding, “C surf”

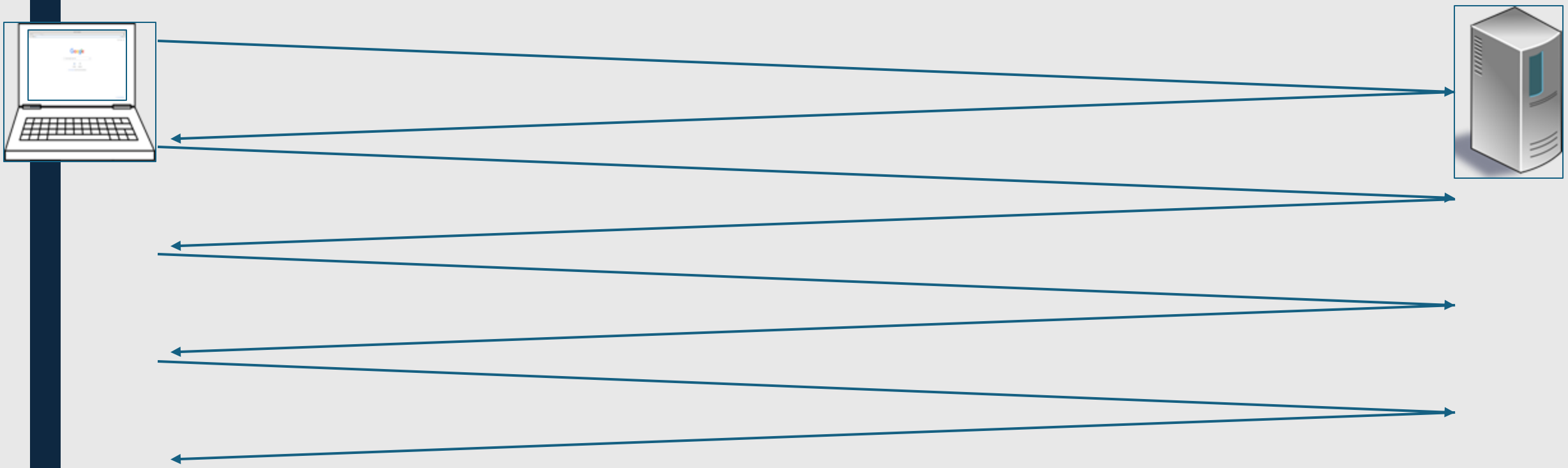
COOKIES

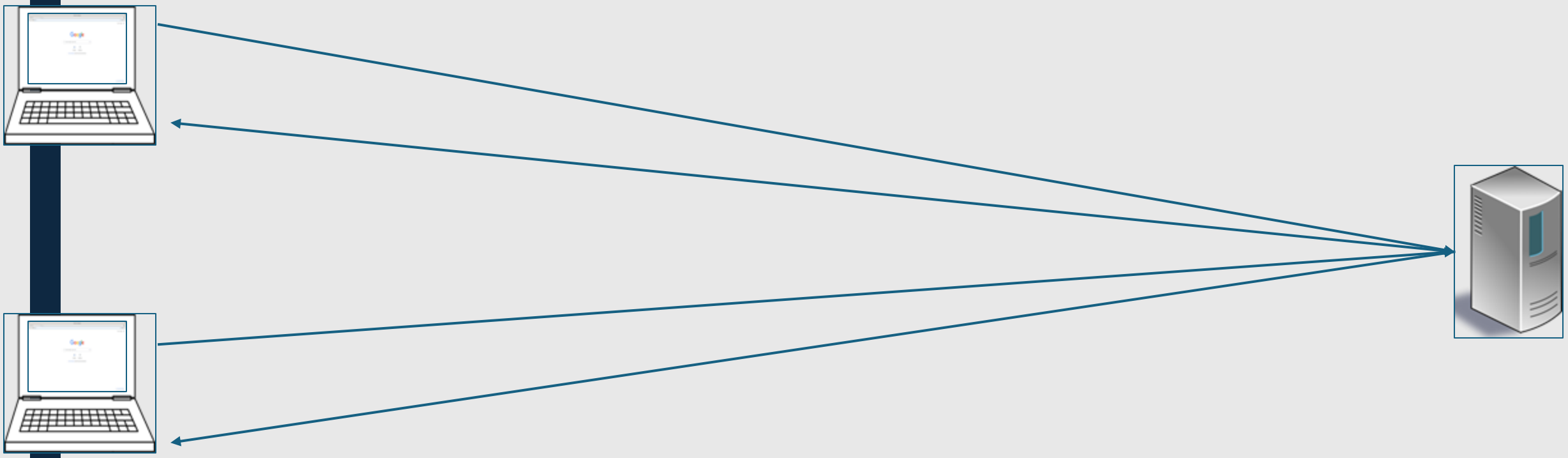
Browser Cookies

Def’n: data strings that enable a stateful browsing session between client and server

E.g., language=english; cartID=ba3fa9dea0;

HTTP is stateless!! So cookies help us keep track of some state ☺





GET www.target.com/s?searchTerm=masks

server sets a cookie to start linking future requests from this client together

HTTP response header contains Set-Cookie keyword followed by the cookie

A cookies just a key-value pair!

HTTP/1.1 200 OK

Set-Cookie: sessionID=9df3ecad22f7
Set-Cookie: userLocation=27599.NC

<html><title>...

GET www.target.com/s?searchTerm=kidsmasks

Cookie:
sessionID=9df3ecad22f7;userLocation=27599.NC

By default, cookies

- can be accessed by JavaScript
- are sent in the clear
- are returned to the requesting server directory

By default, cookies

- can be accessed by JavaScript
- are sent in the clear
- are returned to the requesting server directory

URI setting the cookie:	Cookie returned to:	Cookie not returned to:
sub.site.com/dir/file	sub.site.com/dir/fileA sub.site.com/dir/subdir/fileB	sub.site.com bub.site.com site.com

Cookie WS Problems

Threat Model

- Attacker is not “in the browser”
- Attacker is not eavesdropping on client--server communication
- Attacker can induce victim to click on a link

Alice

bank.com

1. Alice logs into bank.com
2. bank sends session ID to Alice's browser

Alice

bank.com

1. Alice logs into bank.com
2. bank sends session ID to Alice's browser

```
GET http://bank.com/fundxfer.php?to=csturton&value=2500 HTTP/1.1

Cookie: sessionID=2hdkg7s0zwlsl.-S_sfhl25JLLKx
User-Agent: Mozilla/5.0
```

Alice

bank.com

1. Alice logs into bank.com
2. bank sends session ID to Alice's browser

GET http://bank.com/fundxfer.php?to=kakiryan&value=2500 HTTP/1.1

Cookie: sessionID=2hdkg7s0zwlsl.-S_sfhl25JLLKx
User-Agent: Mozilla/5.0

Alice

bank.com

1. Alice logs into bank.com
2. bank sends session ID to Alice's browser

```
GET http://bank.com/fundxfer.php?to=kakiryan&value=2500 HTTP/1.1

Cookie: sessionID=2hdkg7s0zwlsl.-S_sfhl25JLLKx
User-Agent: Mozilla/5.0
```

Possible link locations: on a website Alice visits, embedded in an ad, in an image, in an email or text…

```
<a href="http://bank.com/fundxfer.php?to=kakiryan&value=2500">
Click here … Incredible News!!!
</a>
```

```
<img width="0" height="0" border="0"
src="http://bank.com/fundxfer.php?to=kakiryan&value=2500"/>
```

Notes on CSRF

- Requires that Alice is already logged in to bank.com
- Used for state-changing requests
- Request is indistinguishable from a legitimate request

Notes on CSRF

- Requires that Alice is already logged in to bank.com
- Used for state-changing requests
- Request is indistinguishable from a legitimate request

This is an example of a confused deputy attack!

Not a Defense

- HTTPOnly cookie attribute
- HTTP over TLS (HTTPS)
- Checking the referer header

CSRF Defense: synchronizer tokens

Alice

HTTP/1.1 200 OK

Set-Cookie: sessionID=9df3ecad22f7

<html><body><input type="hidden"
name="csrftoken"
value="bda35f355ffca3fe"/>

bank.com

CSRF Defense: cookie to header

Alice

bank.com

HTTP/1.1 200 OK

Set-Cookie: sessionID=9df3ecad22f7;
csrftoken=bda35f355ffca3fe

<html><body>....

CSRF Defense: cookie to header

Alice

bank.com

POST http://bank.com/fundxferform HTTP/1.1

Cookie: sessionID=9df3ecad22f7; csrftoken=bda35f355ffca3fe
CSRFToken: bda35f355ffca3fe
User-Agent: Mozilla/5.0

....

CSRF Defense: double submitting cookies

Alice

bank.com

HTTP/1.1 200 OK

Set-Cookie: sessionID=9df3ecad22f7;
csrftoken=bda35f355ffca3fe

<html><body><input type="hidden"
name="csrftoken"
value="bda35f355ffca3fe"/>