CSE467: Computer Security

13. Introduction & Web Programming

Seongil Wi

Department of Computer Science and Engineering

Modification on Previous Lecture Slide (Before)

Do Not Read Uninitialized Memory

```
void set_flag(int number, int *sign_flag) {
  if (NULL == sign_flag) return;
  if (number > 0) {
    *sign_flag = 1;
  } else if (number < 0) {
    *sign_flag = -1;
  }
}
int is_negative(int number) {
  int sign;
  set_flag(number, &sign);
  return sign < 0;
}
```

Modification on Previous Lecture Slide (after)

3

Do Not Read Uninitialized Memory

47

```
void set_flag(int number, int *sign_flag) {
  if (NULL == sign_flag) return;
  if (number > 0) {
    *sign_flag = 1;
  } else if (number < 0) {
    *sign_flag = -1;
  }
}
int is_negative(int number) {
  int sign;
  set_flag(number, &sign);
  return sign < 0;
}
```

What if number is 0?
sign_flag is not initialized

Notice

- There will be Q&A session for HW2 after the class

Introduction to Web Security

The Web has won

- Used by billions of people to store/retrieve information

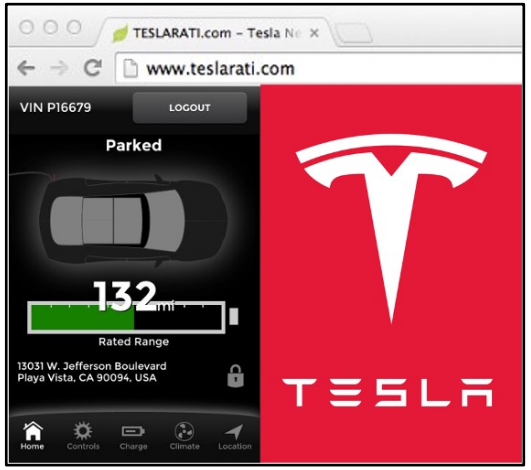
2B users monthly

2.3M searches per second

- Large coverage in desktop/mobile application

WebView

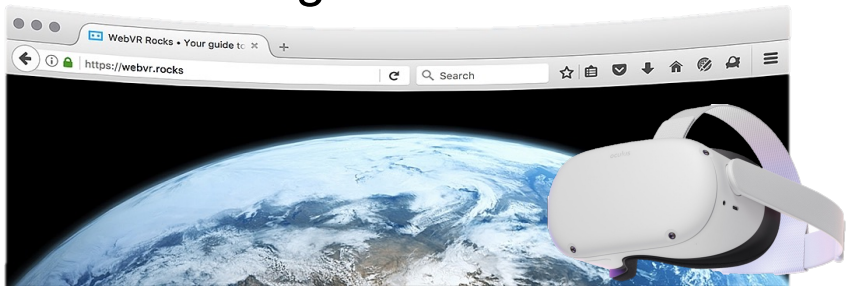
- User interface for emerging systems



Self-driving car

Cryptocurrency

AI models



WebVR

… and the hackers with it

- Used by billions of people to store/retrieve information

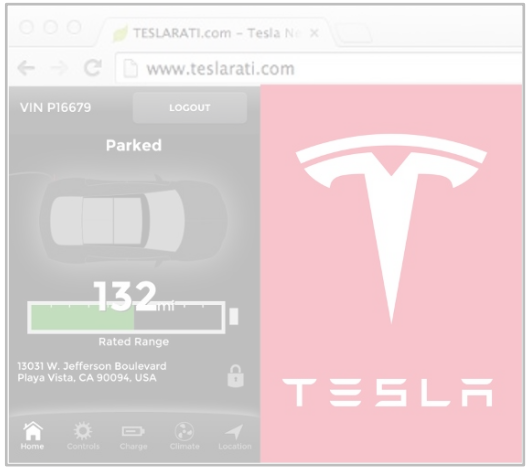
2B users monthly

2.3M searches per second

- Large coverage in desktop/mobile application

WebView

- Interface for emerging systems

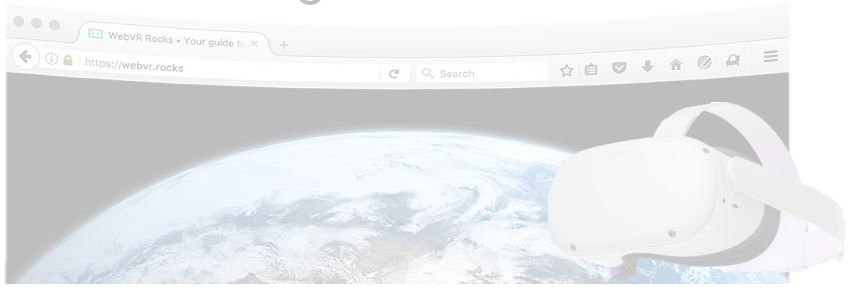


Self-driving car

Cryptocurrency

OpenAI ChatGPT

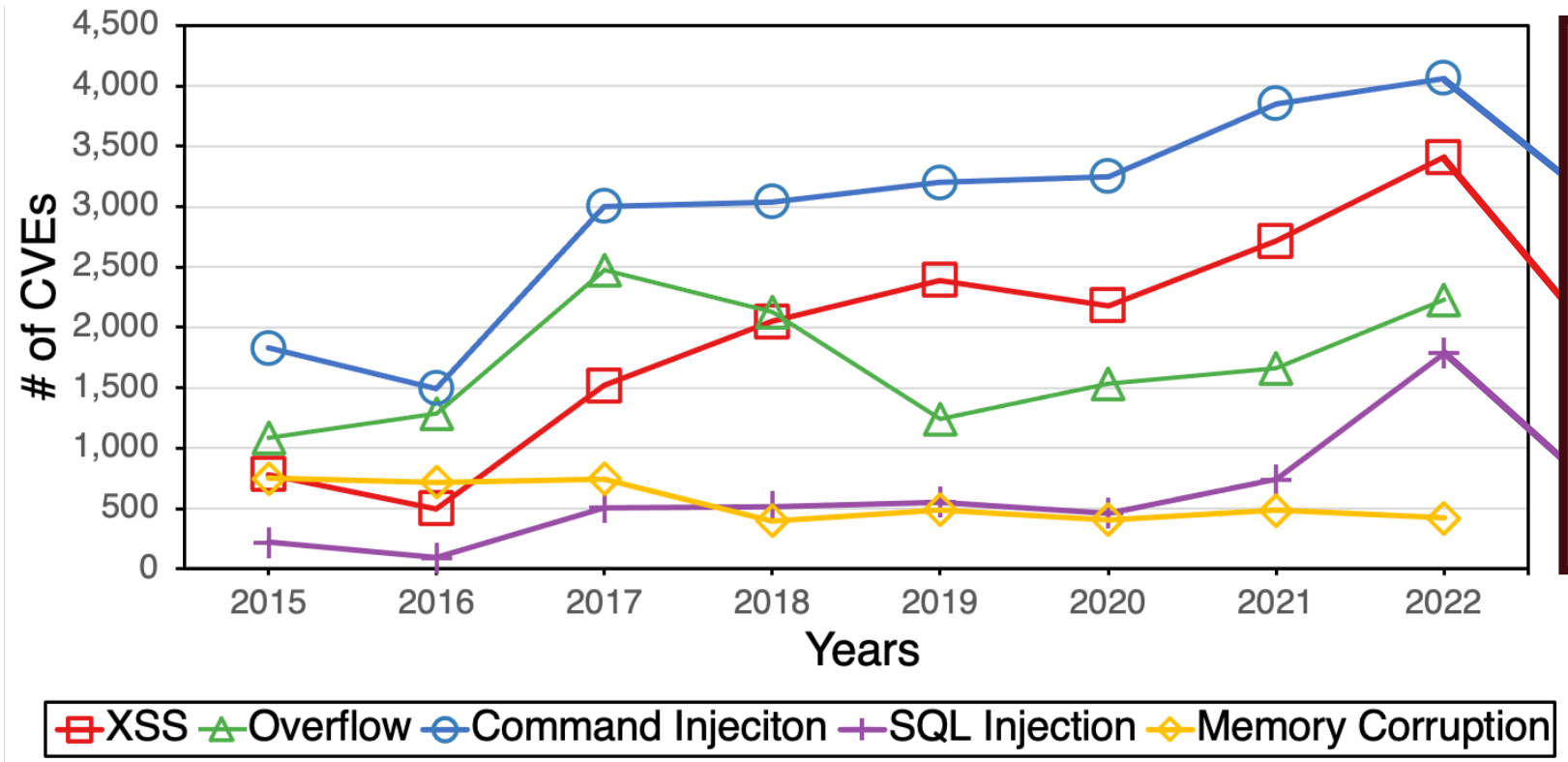
AI models



WebVR

Why Web Security?

- *Web attacks* accounted for **48.6%** of all reported threats



Web attacks

- Initiate denial-of-service (DoS) attacks
- Access to sensitive information
- Enable remote code execution

https://www.cvedetails.com/vulnerabilities-by-types.php

… and the hackers with it

Crypto

2FA compromise led to $34M Crypto.com hack

Anita Ramaswamy @anitaramaswamy / 3:13 AM GMT+9 • January 21, 2022 Comment

Jonathan Greig
January 17th, 2023

Briefs Cybercrime

Norton LifeLock says 925,000 accounts targeted by credential-stuffing attacks

Web skimming hackers infiltrate over 40 ecommerce websites - that we know of

News By Abigail Opiah published December 08, 2022

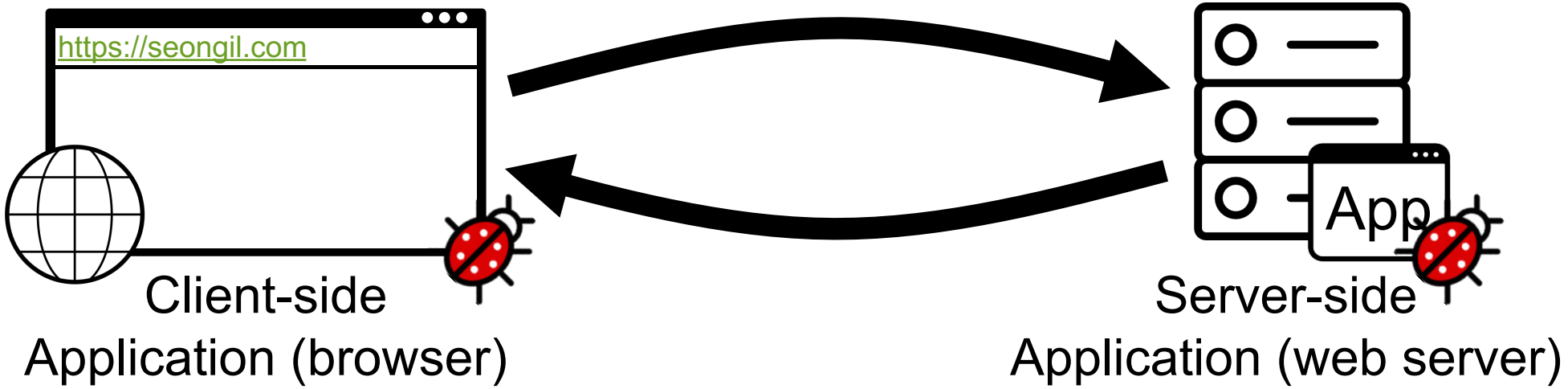
Companies paid $4.2M bug bounties

Web threats are critical!

Introduction to Web Security

- We are going to study and discuss the web attacks and defenses.

- Web Programming Basic
- Server-side Web Attacks & Defenses
- Client-side Web Attacks & Defenses

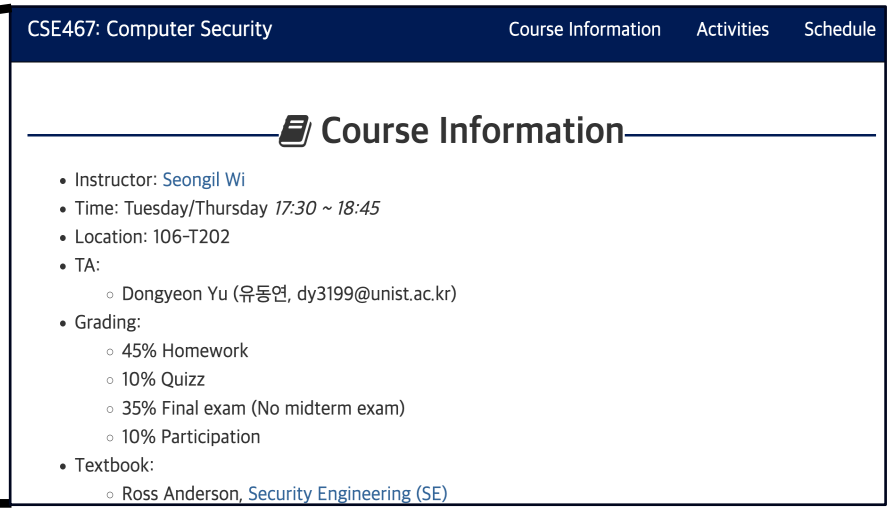


Web Programming Basic

Hypertext Markup Language (HTML)

- Markup language for web page layout
  - NOT programming language (i.e., for computation)!
- A web page (document) is written in HTML using markup tags
  - E.g., `<p>`, `<img>`

```
<html>
  <body>
    ...
  </body>
</html>
```



Hypertext Markup Language (HTML)

- Markup language for web page layout
  - NOT programming language (i.e., for computation)!
- A web page (document) is written in HTML using markup tags
  - E.g., `<p>`, `<img>`
- A **browser** interprets a web page when rendering the page
- Describes a hyper-text document
  - E.g., image, audio, video

```
<html>
  <body>
    . . .
  </body>
</html>
```

What if we need computation?
⇒ JavaScript!

Uniform Resource Locators (URLs)

- Global identifiers of network-retrievable documents
- Example

http://websec-lab.com:80/cse467?id=1223#homework

Uniform Resource Locators (URLs)

- Global identifiers of network-retrievable documents
- Example

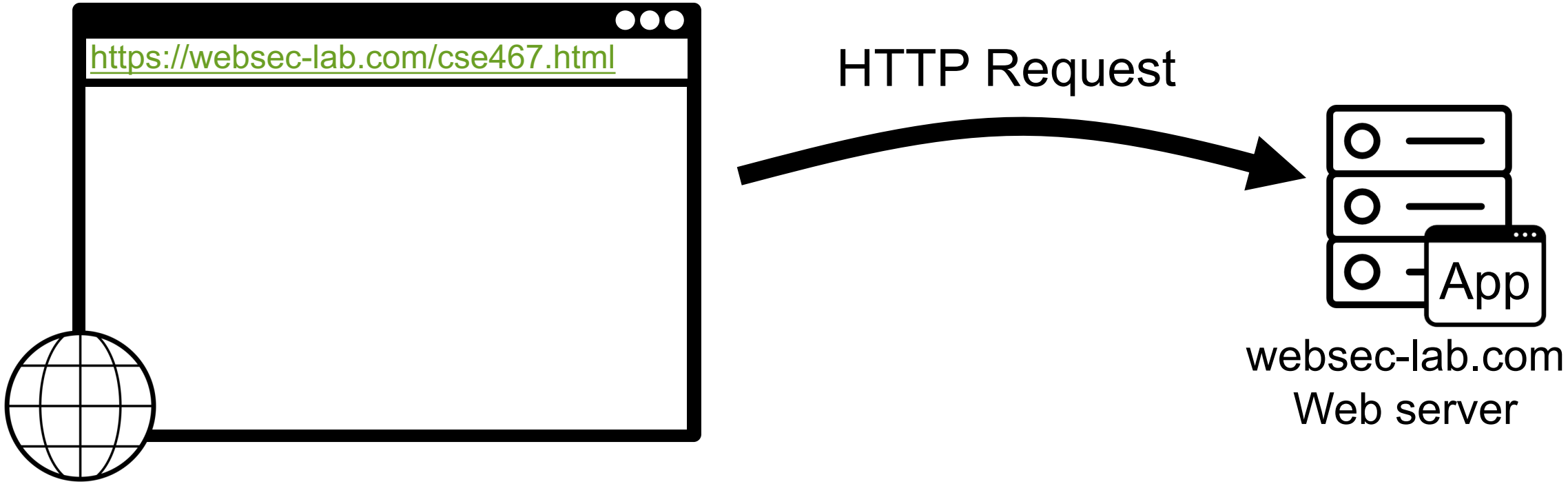
http://websec-lab.com:80/cse467?id=1223#homework

Protocol (http, https, …) | Hostname | Port | Path | Query | Fragment

- Special characters are encoded as hex:
  - New line → %0A
  - Space → %20
  - + → %2B

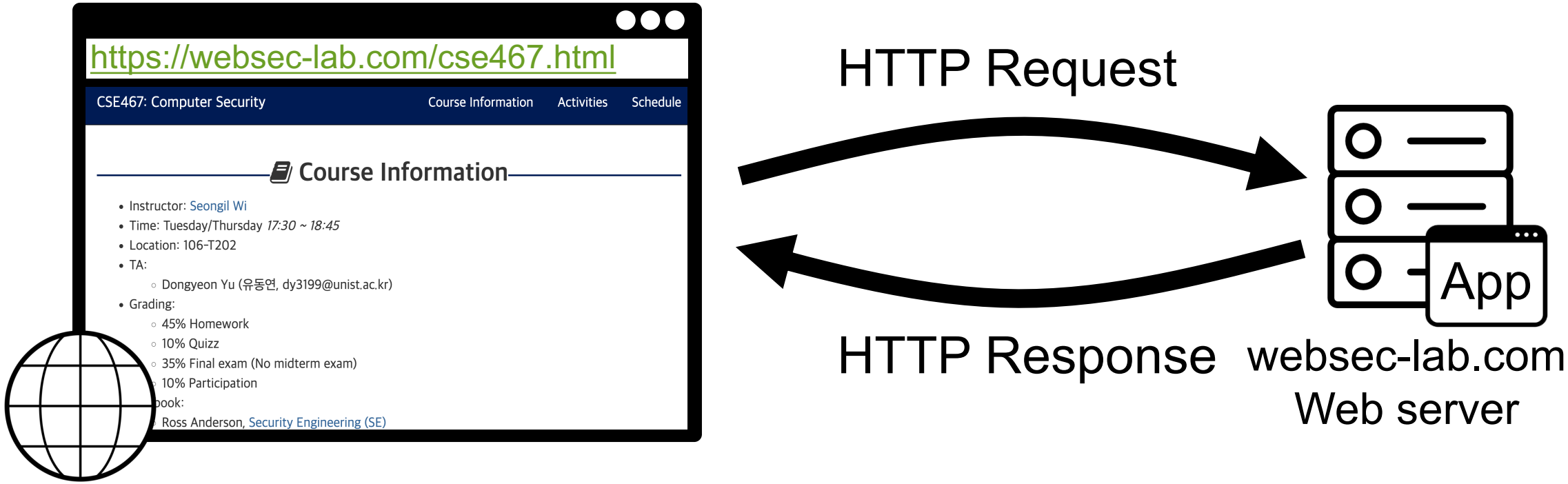
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



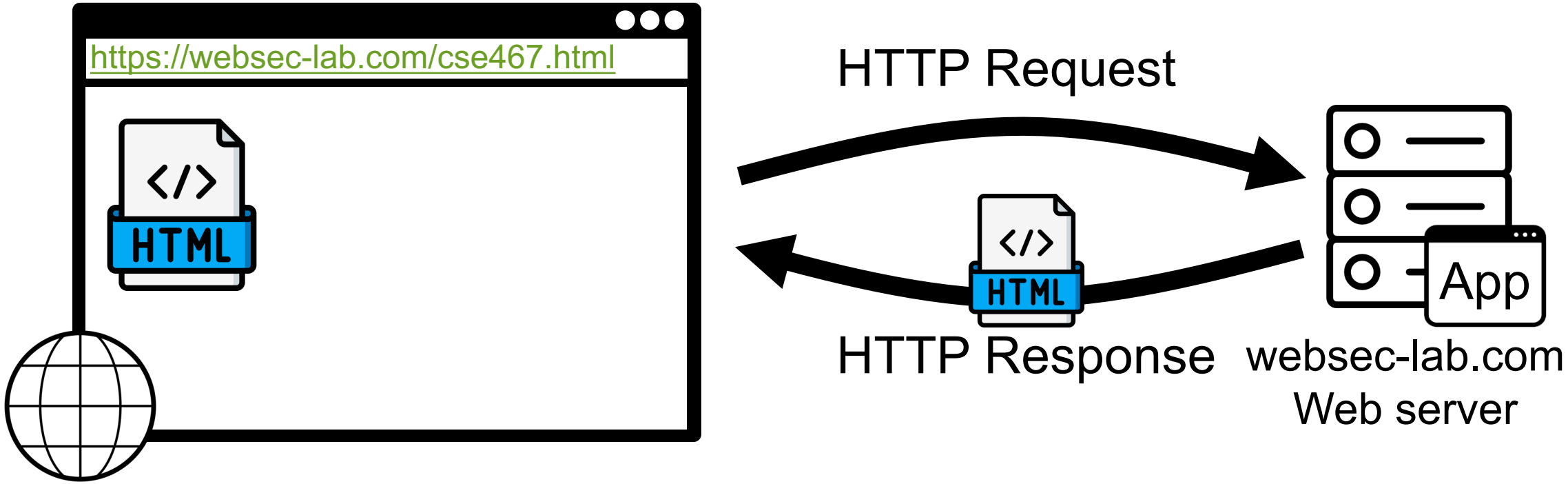
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



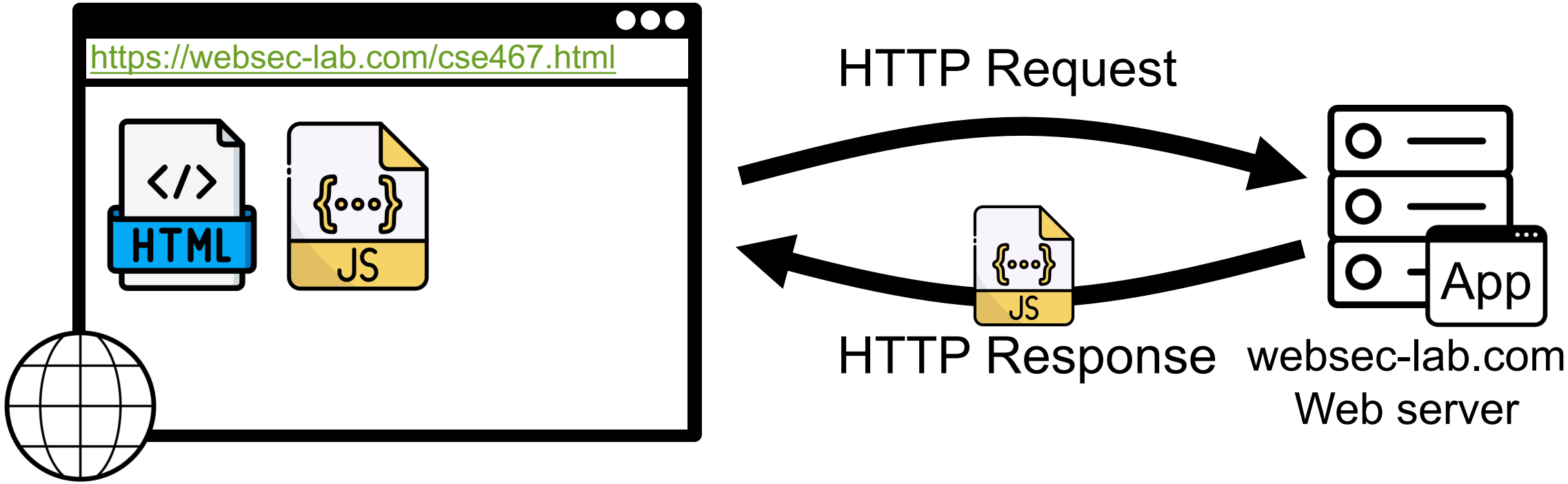
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



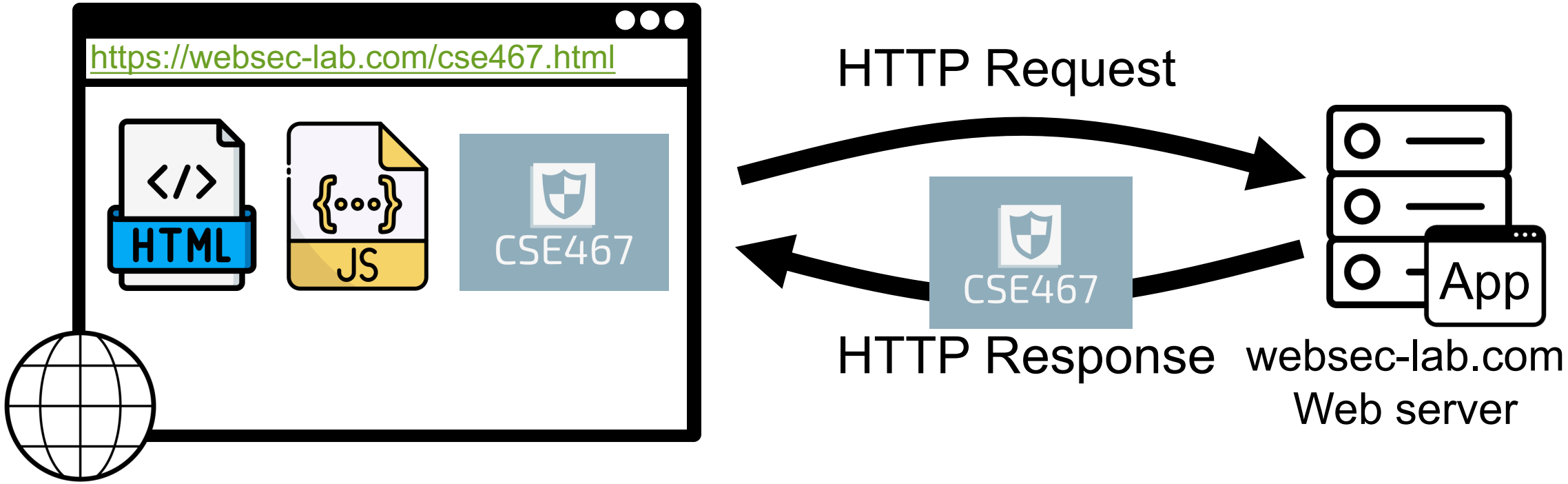
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



HTTP Request

```
GET  /cse467.html  HTTP/1.1
Host: websec-lab.com
Accept-Language: en
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;)
Referer: http://google.com
```

HTTP Request

Method | File path | Protocol

Request Line

```
GET  /cse467.html  HTTP/1.1
Host: websec-lab.com
Accept-Language: en
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;)
Referer: http://google.com
```

Many HTTP Methods

- **GET**: Get the resource at the specified URL
- **POST**: Create new resource at URL with payload
- **PUT**: Replace current representation of the target resource with request payload
- **PATCH**: Update part of the resource
- **DELETE**: Delete the specified URL

HTTP Request

Method

File path

Protocol

List of HTTP headers, as key-value pair

Request Line

Request headers

Body (empty)

```
GET  /cse467.html  HTTP/1.1
Host: websec-lab.com
Accept-Language: en
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;)
Referer: http://google.com
```

Contain the address from which a resource has been requested

Referrer Header

http://google.com

Google

CSE467: Computer Security

전체 이미지 지도 뉴스 쇼핑

검색결과 약 5개 (0.28초)

GitHub
https://websec-lab.github.io › courses › 2023f-cse467

CSE467: Computer Security

Course **Information**. Instructor: **Seongil Wi**; Time: Tuesc
T202; TA: Dongyeon Yu (유동연, dy3199@**unist**.ac.kr). Gr

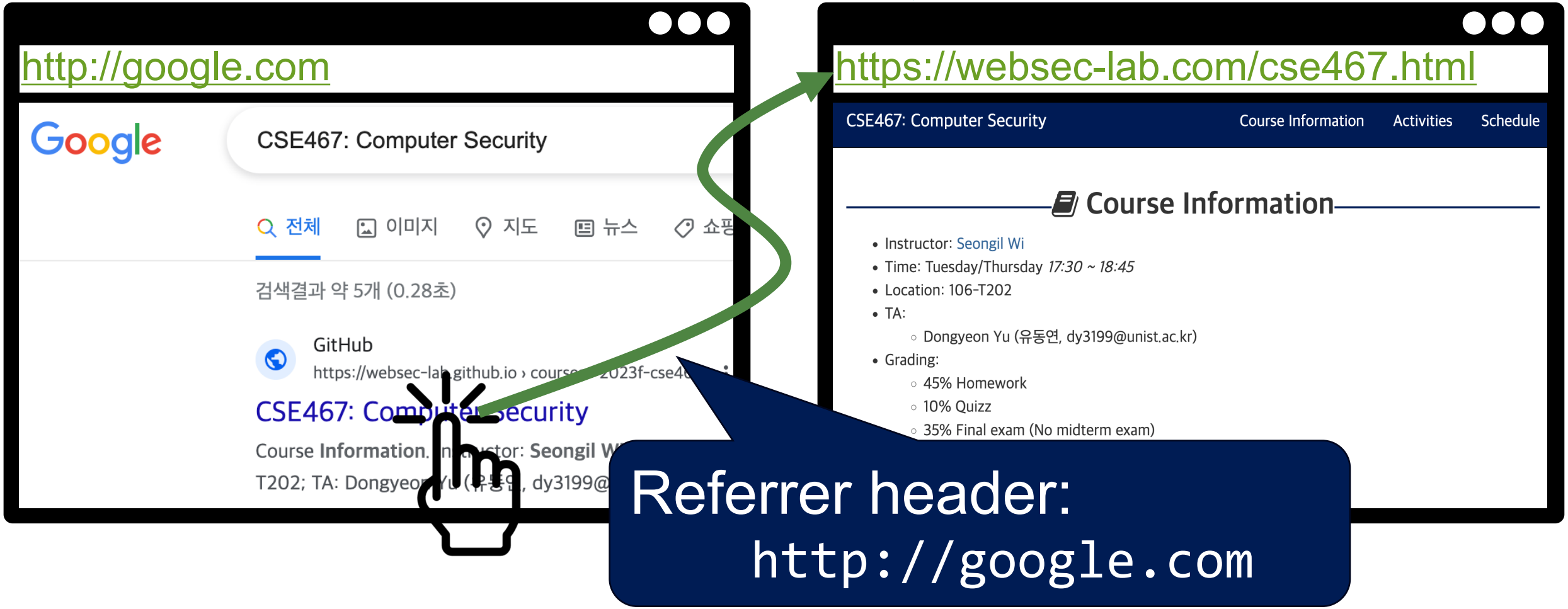
https://websec-lab.com/cse467.html

CSE467: Computer Security | Course Information | Activities | Schedule

Course Information

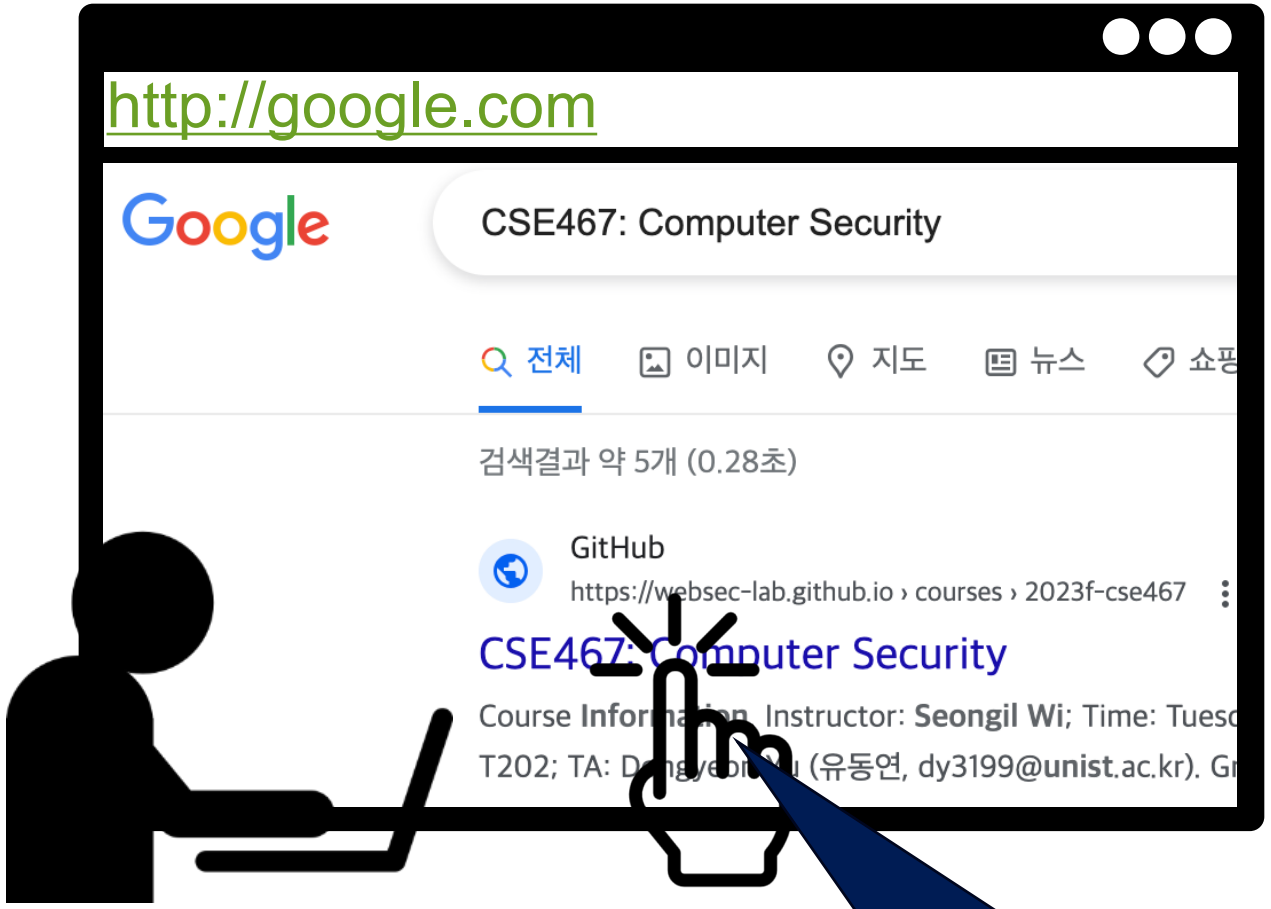
- Instructor: Seongil Wi
- Time: Tuesday/Thursday *17:30 ~ 18:45*
- Location: 106-T202
- TA:
  - Dongyeon Yu (유동연, dy3199@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - Ross Anderson, Security Engineering (SE)

Referrer Header



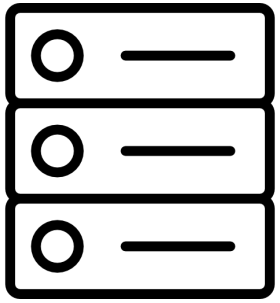
Referrer header:
`http://google.com`

Referrer Header in Detail



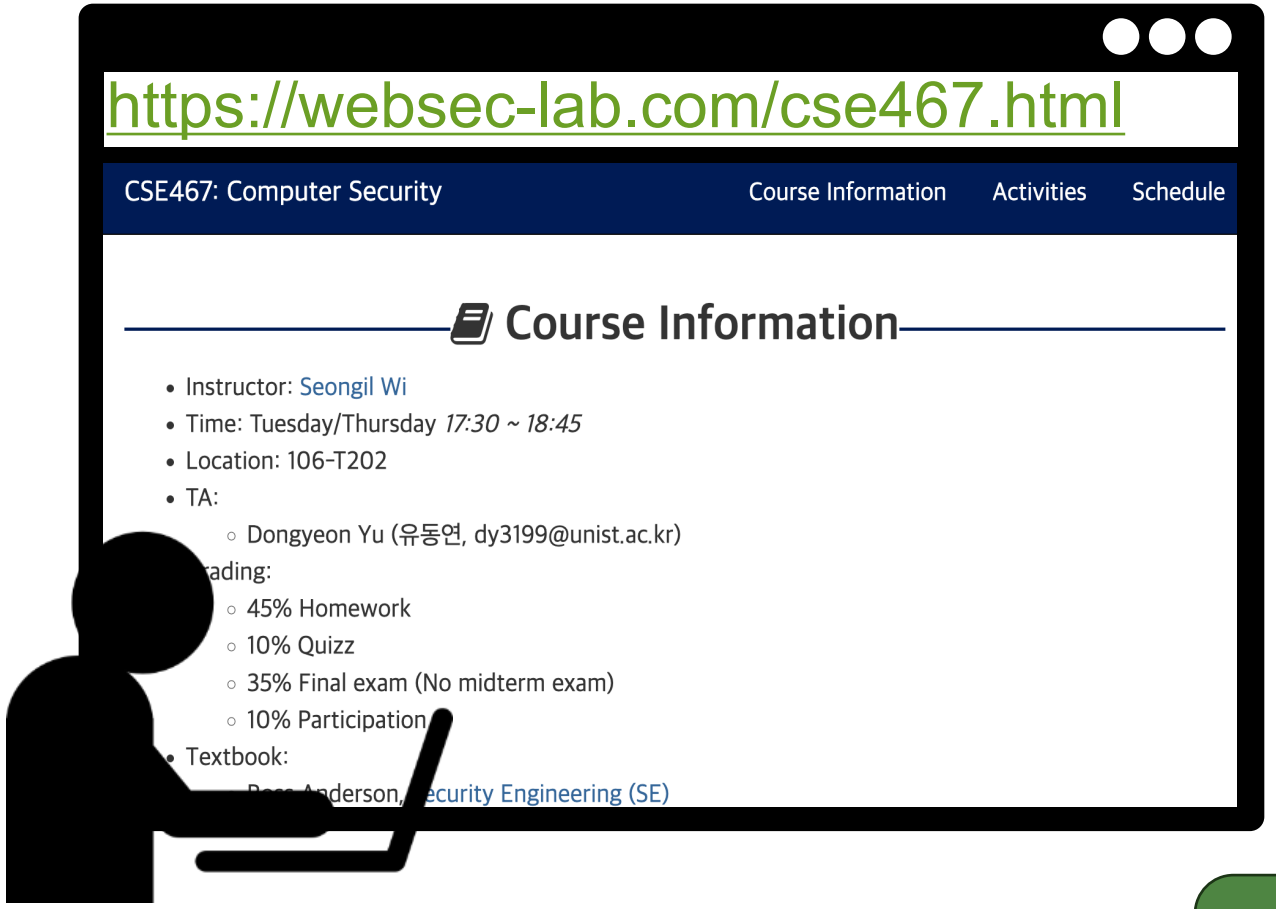
(1) Click the link

(2) Send HTTP request (with referrer header)

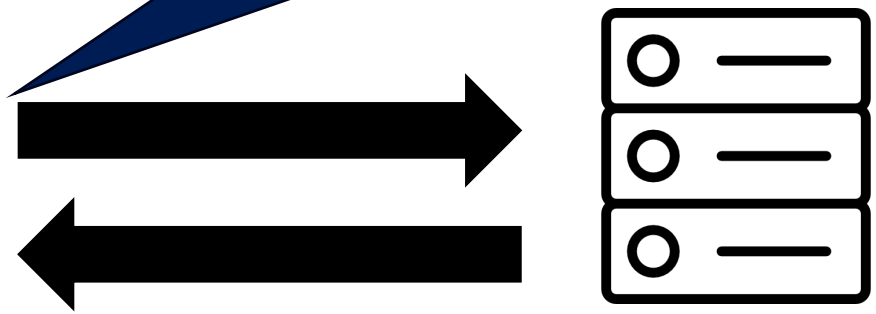


Web server
websec-lab.github.io

Referrer Header in Detail



(2) Send HTTP request (with referrer header)



Web server
websec-lab.github.io

The server can analyze **where the request originated**

Question

Are there any security issues with the referrer header?

Referrer Entails a Loss of Privacy

Sensitive information

http://samsung.com/[project_name]

Website for secret projects

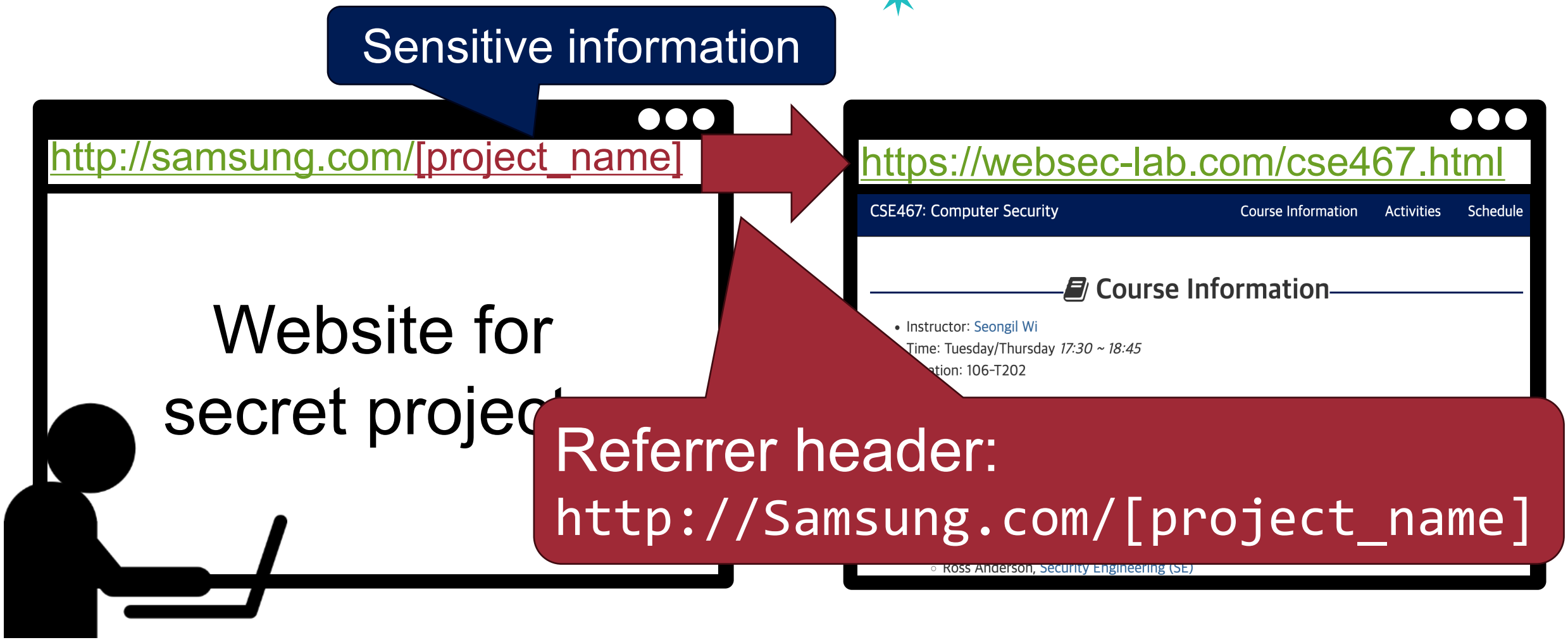
https://websec-lab.com/cse467.html

CSE467: Computer Security | Course Information | Activities | Schedule

Course Information

- Instructor: Seongil Wi
- Time: Tuesday/Thursday *17:30 ~ 18:45*
- Location: 106-T202
- TA:
  - Dongyeon Yu (유동연, dy3199@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - Ross Anderson, Security Engineering (SE)

Referrer Entails a Loss of Privacy



Referrer Entails a Loss of Privacy

HTTP Response

Status Line

Response headers

Response body

```
HTTP/1.1  200 OK
Date: Sat, 21 Oct 2023 07:58:24 GMT
Connection: Keep-alive
Content-Type: text/html
Content-Length: 2543

<html>
  <body>
    some data...
  </body>
</html>
```

HTTP Response

HTTP version | Status code | Status text

Status Line

Response headers

Response body

```
HTTP/1.1  200 OK
Date: Sat, 21 Oct 2023 07:58:24 G
Connection: Keep-alive
Content-Type: text/html
Content-Length: 2543

<html>
  <body>
    some data...
  </body>
</html>
```

HTTP STATUS CODES

2xx Success

200	Success / OK

3xx Redirection

301	Permanent Redirect
302	Temporary Redirect
304	Not Modified

4xx Client Error

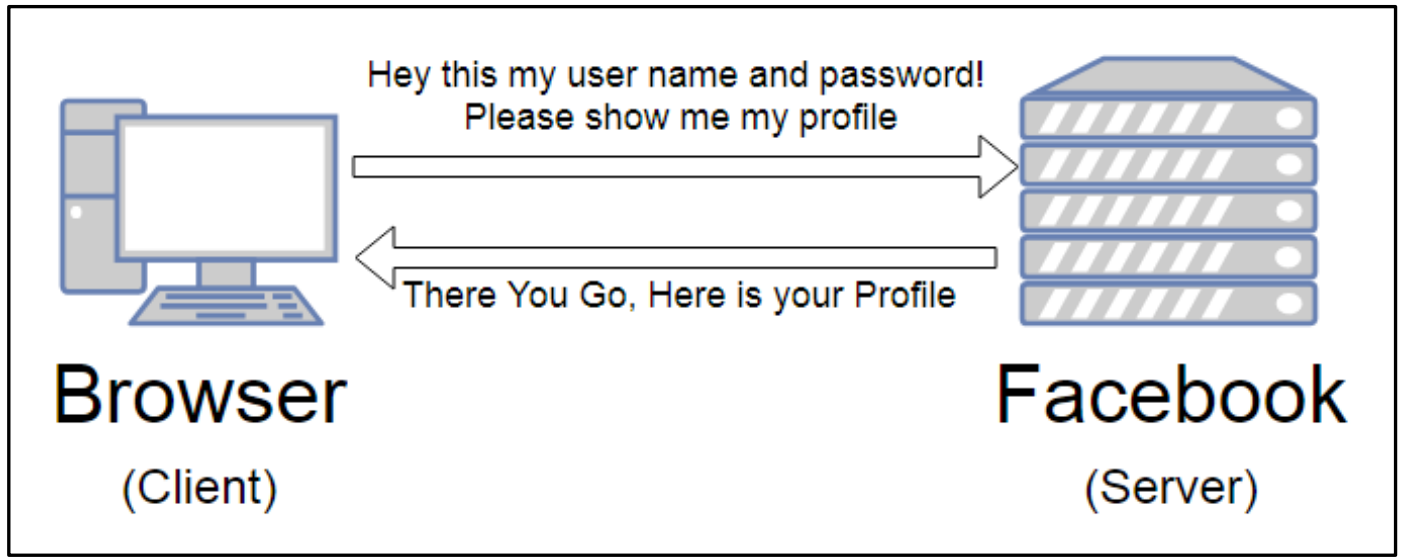
401	Unauthorized Error
403	Forbidden
404	Not Found
405	Method Not Allowed

5xx Server Error

501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

HTTP is a Stateless Protocol

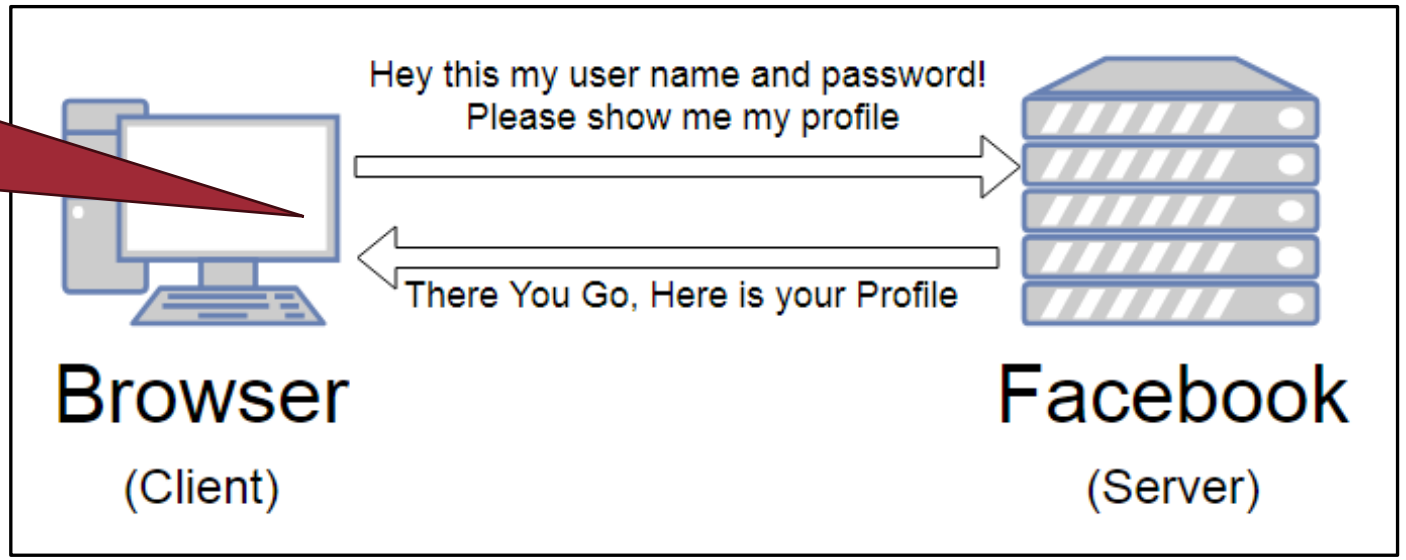
1st try



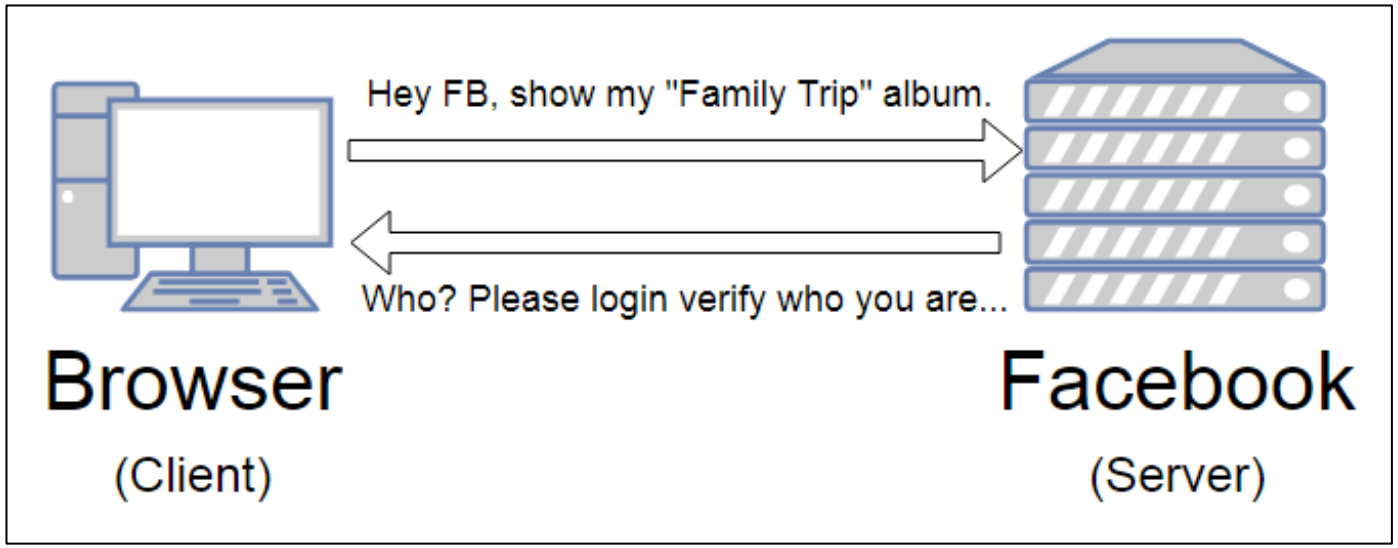
HTTP is a Stateless Protocol

Stateless protocol:
Each request is independent to previous request

1st try



2nd try



How to make HTTP "act" stateful?

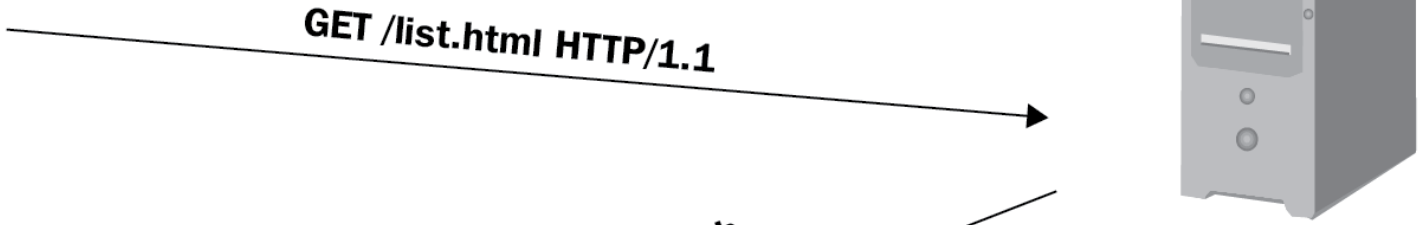
Cookie: Making HTTP Stateful

- HTTP cookie: small piece of data that a server sends to the browser, who stores it and sends it back with subsequent requests

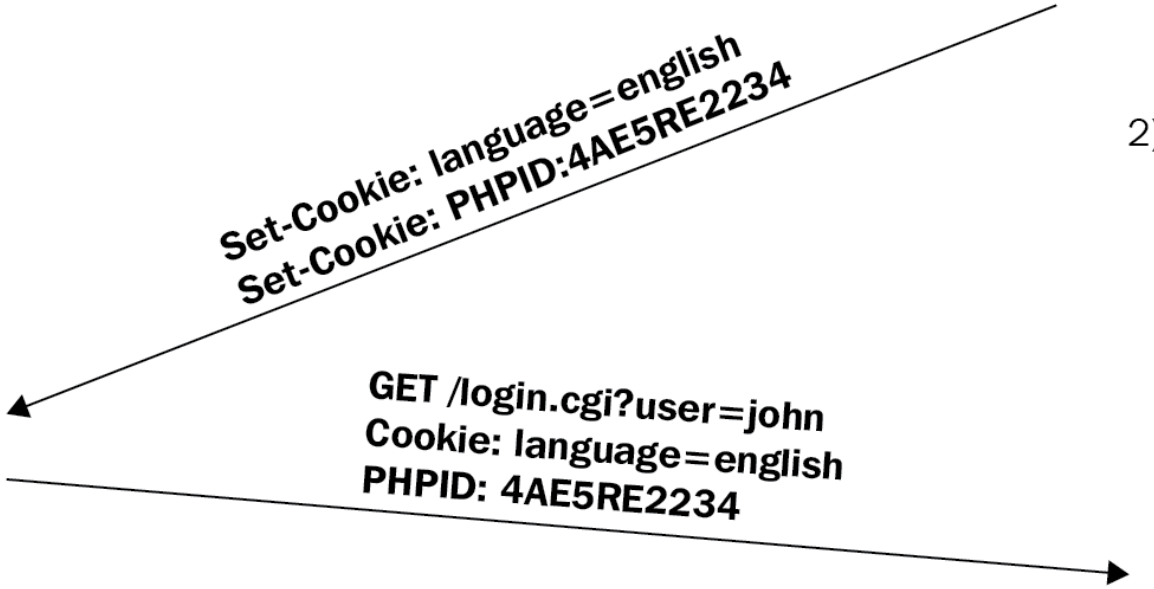
- What is this useful for?
  - Session management: logins, shopping carts, etc.
  - Personalization: user preferences, themes, etc.
  - Tracking: recording and analyzing user behavior

Cookie: Making HTTP Stateful

1) Browser sends request



2) Server sets a cookie



3) Browser sends cookie back in subsequent requests

JavaScript (JS)

- Developed by Brendan Eich at Netscape
- Later standardized for browser compatibility
  - ECMAScript Edition 3 (a.k.a., JavaScript 1.5)

- HTML may contain JS program code to make web pages more dynamic

JS Example (1)

```
<html>
  <p id="demo"></p>
  <script>
    document.getElementById("demo").
                innerHTML = 5 + 6;
  </script>
</html>
```

https://websec-lab.github.io/courses/2023f-cse467/demo/demo1.html

JS Example (1)

Inline script with script tag

```
<html>
  <p id="demo"></p>
  <script>
    document.getElementById("demo").
                 innerHTML = 5 + 6;
  </script>
</html>
```

https://websec-lab.github.io/courses/2023f-cse467/demo/demo1.html

JS Example (2)

```
<html>
  <button type="button" onclick="document.write(5 + 6)">
    Try it
  </button>
</html>
```

https://websec-lab.github.io/courses/2023f-cse467/demo/demo2.html

JS Example (2)

When the button is clicked,
overwrite whole document with 11

```
<html>
  <button type="button" onclick="document.write(5 + 6)">
    Try it
  </button>
</html>
```

Inline script with
`onclick` event handler

https://websec-lab.github.io/courses/2023f-cse467/demo/demo2.html

JS Example (3)

index.html

```
<html>
  <script src="write.js">
  </script>
</html>
```

write.js

```
document.write(5 + 6)
```

https://websec-lab.github.io/courses/2023f-cse467/demo/demo3.html

JS Example (3)

Overwrite whole document with 11

index.html

```
<html>
  <script src="write.js">
  </script>
</html>
```

write.js

```
document.write(5 + 6)
```

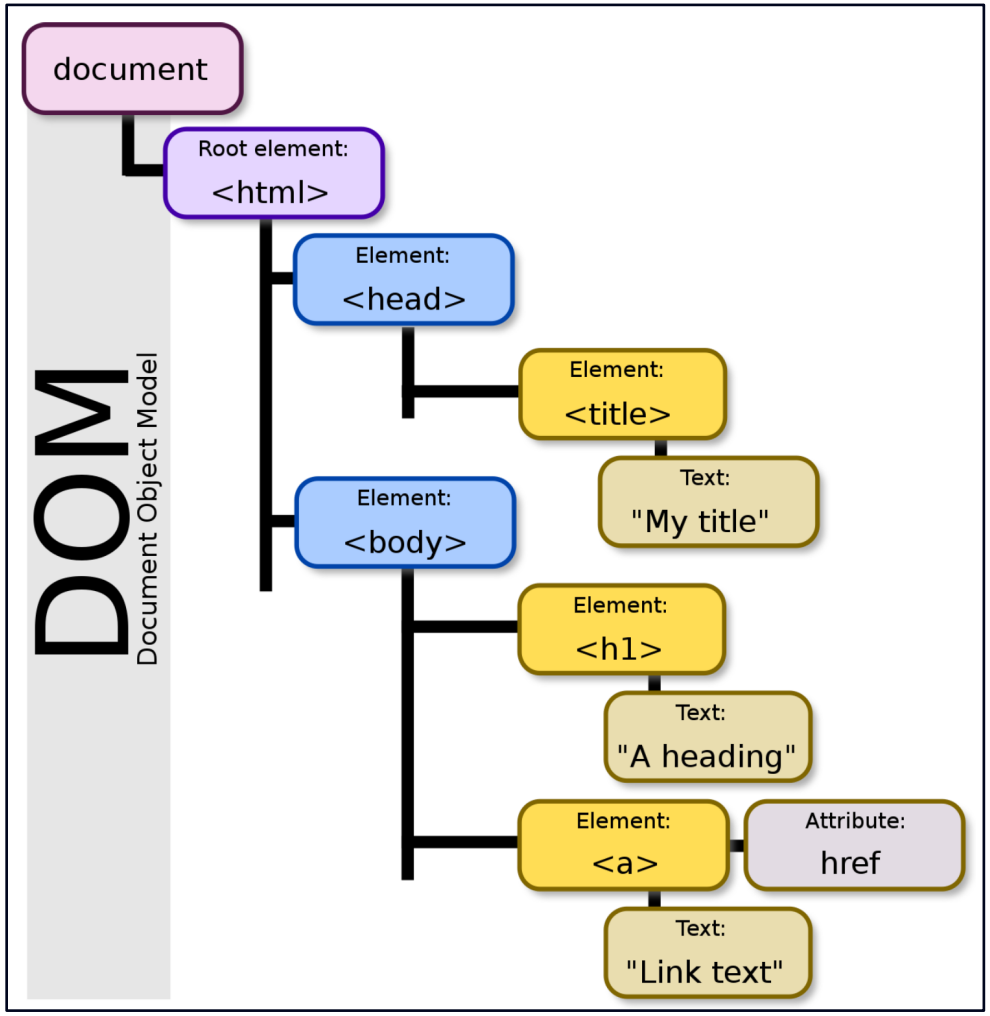
External script with `src` attribute

https://websec-lab.github.io/courses/2023f-cse467/demo/demo3.html

Document Object Model (DOM)

- An HTML document: structured data

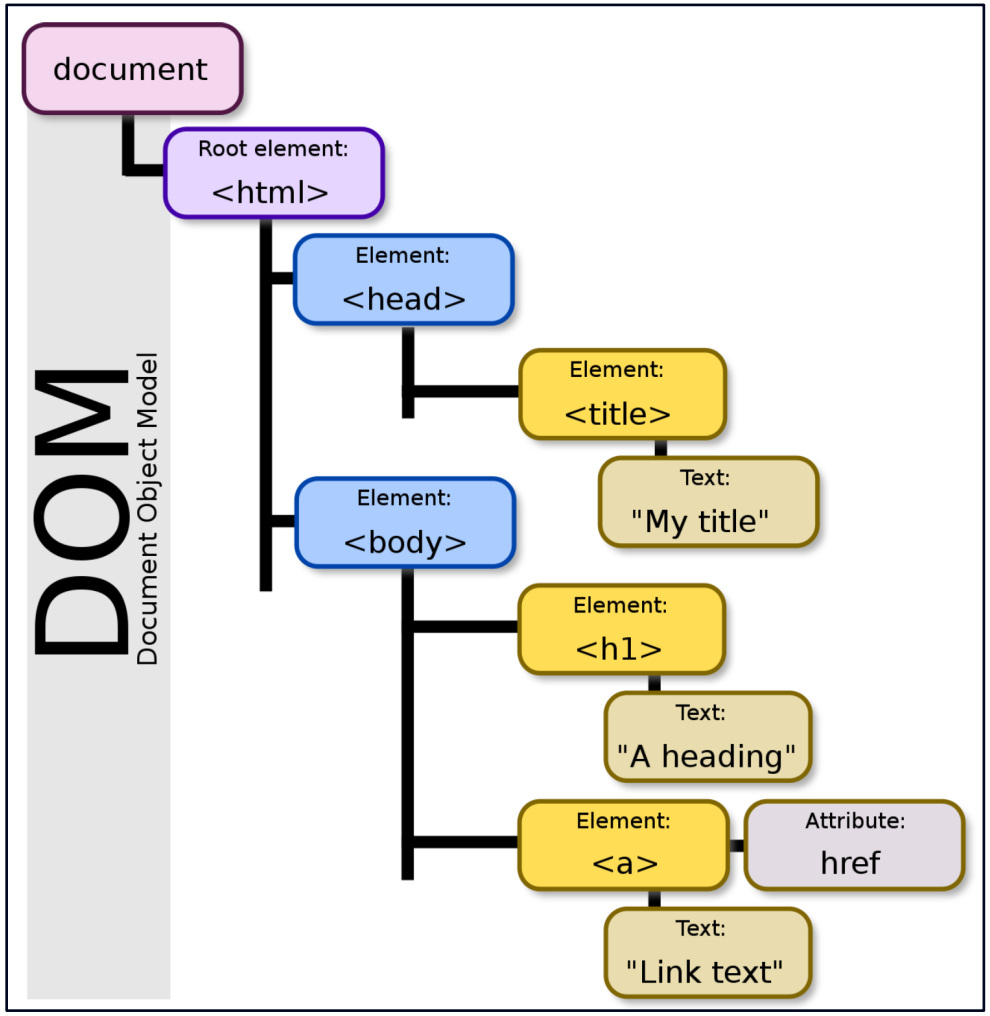
```
<html>
  <head>
    <title>
      My title
    </title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="cse467.com">Link text</a>
  </body>
<body>
```



Document Object Model (DOM)

- An HTML document: structured data
- Can be modified by JavaScript

```
<html>
  <head>
    <title>
      My title
    </title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="cse467.com">Link text</a>
  </body>
<body>
```



Changing HTML DOM using JS

- JavaScript can change all the HTML DOM components in the page!

- using several APIs
  - `createElement(elementName)`
  - `createTextNode(text)`
  - `appendChild(newChild)`
  - `removeChild(node)`

Changing HTML DOM using JS (Example)


```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

- Item 1

Changing HTML DOM using JS (Example)

```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

- Item 1

```
<script>
  var list = document.getElementById('t1')
  var newitem = document.createElement('li')
  var newtext = document.createTextNode('Item 2')
  list.appendChild(newitem)
  newitem.appendChild(newtext)
</script>
```

Changing HTML DOM using JS (Example)

```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

- Item 1
- Item 2

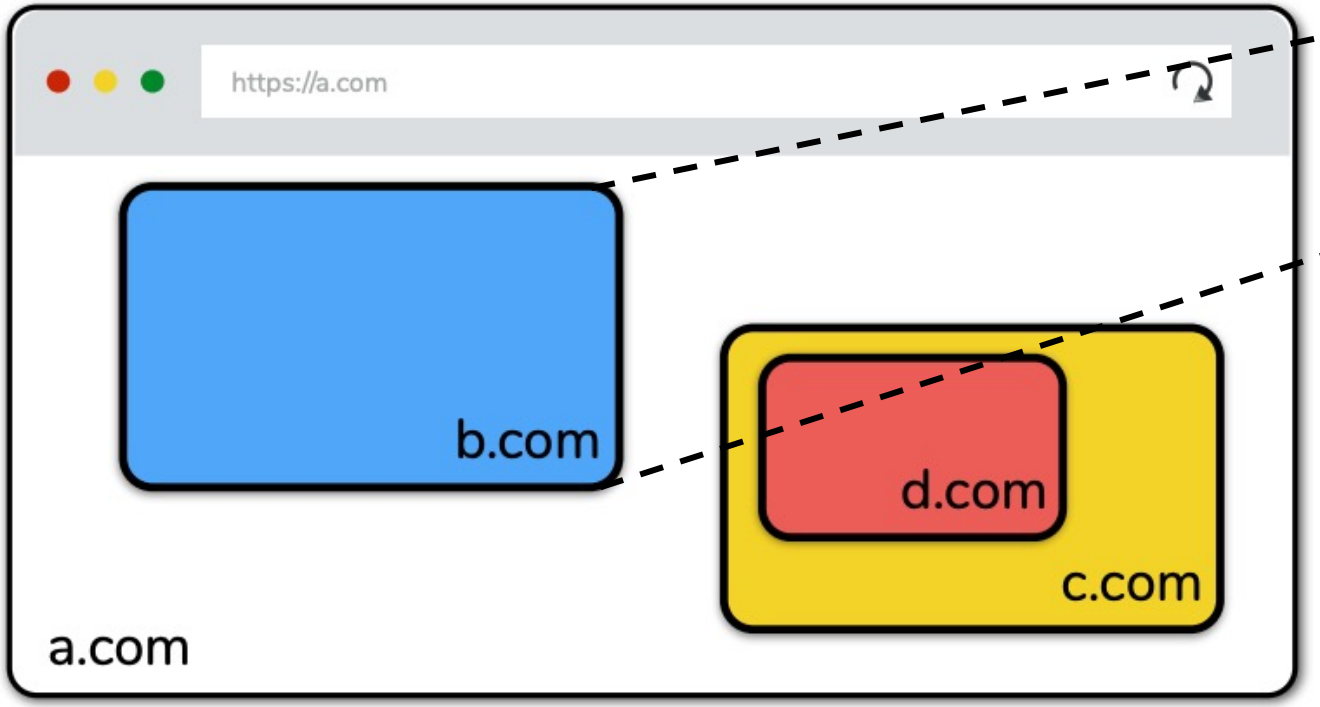
```
<script>
  var list = document.getElementById('t1')
  var newitem = document.createElement('li')
  var newtext = document.createTextNode('Item 2')
  list.appendChild(newitem)
  newitem.appendChild(newtext)
</script>
```

Basic Browser Execution Model

- Each browser window…
  - Loads content
  - Parses HTML and runs JavaScript
  - Fetches sub resources (e.g., images, CSS, Javascript)
  - Respond to events like `onClick`, `onMouseover`, `onLoad`, `setTimeout`

Nested Execution Model

- Windows may contain frames from different sources
  - **Frame**: rigid visible division
  - **iFrame**: floating inline frame




```
<iframe src="b.com">
</iframe>
```

Nested Execution Model

- Windows may contain frames from different sources
  - **Frame**: rigid visible division
  - **iFrame**: floating inline frame

- Why use frames?
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent may work even if frame is broken

Web Threat Models

- **Network attacker**
- **Remote attacker**
- **Web attacker**

Web Threat Models

- **Network attacker**: resides somewhere in the communication link between client and server
  - Passive: evasdropping
  - Active: modification of messages, replay…

- **Remote attacker**

- **Web attacker**

Web Threat Models

- **Network attacker**: resides somewhere in the communication link between client and server
  - Passive: evasdropping
  - Active: modification of messages, replay…

- **Remote attacker:** can connect to remote system via the network
  - Mostly targets the server

- **Web attacker**

Web Threat Models

- **Network attacker**: resides somewhere in the communication link between client and server
  - Passive: evasdropping
  - Active: modification of messages, replay…

- **Remote attacker:** can connect to remote system via the network
  - Mostly targets the server

- **Web attacker**: controls attacker.com
  - Can obtain SSL/TLS certificates for attacker.com
  - Users can visit attacker.com

Web Attacker

Victims can visit attacker's webpage

Web attacker can control of his webpage

Victim

Web attacker

Question

- Is the ***web attacker*** has a control on the victim's referrer header?

Question?