CSE467: Computer Security

8. Introduction & Web Programming

Seongil Wi

Department of Computer Science and Engineering

Recap: Hack Class101

- Find unknown security issues on Class101 websites!
- Instruction: https://bounty.class101.net/
  - Foreigners should use a translator
- Activity period: 03/03 ~ 06/18
- DO NOT try anything illegal!

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 o le to store/retrieve inf on

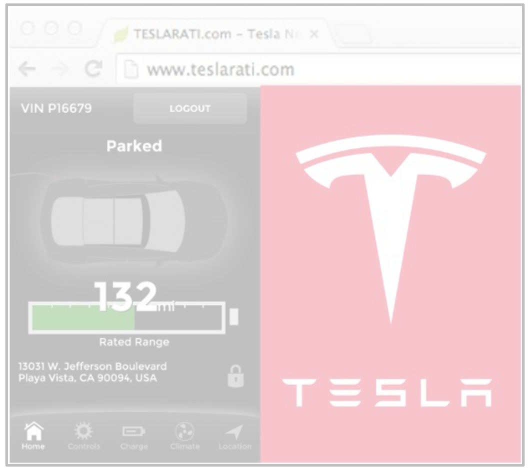
2B users monthly

2.3M searches per second

- Large coverage in desktop/mobile ap on

WebView

erface for ging systems



Self-driving car

Cryptocurrency

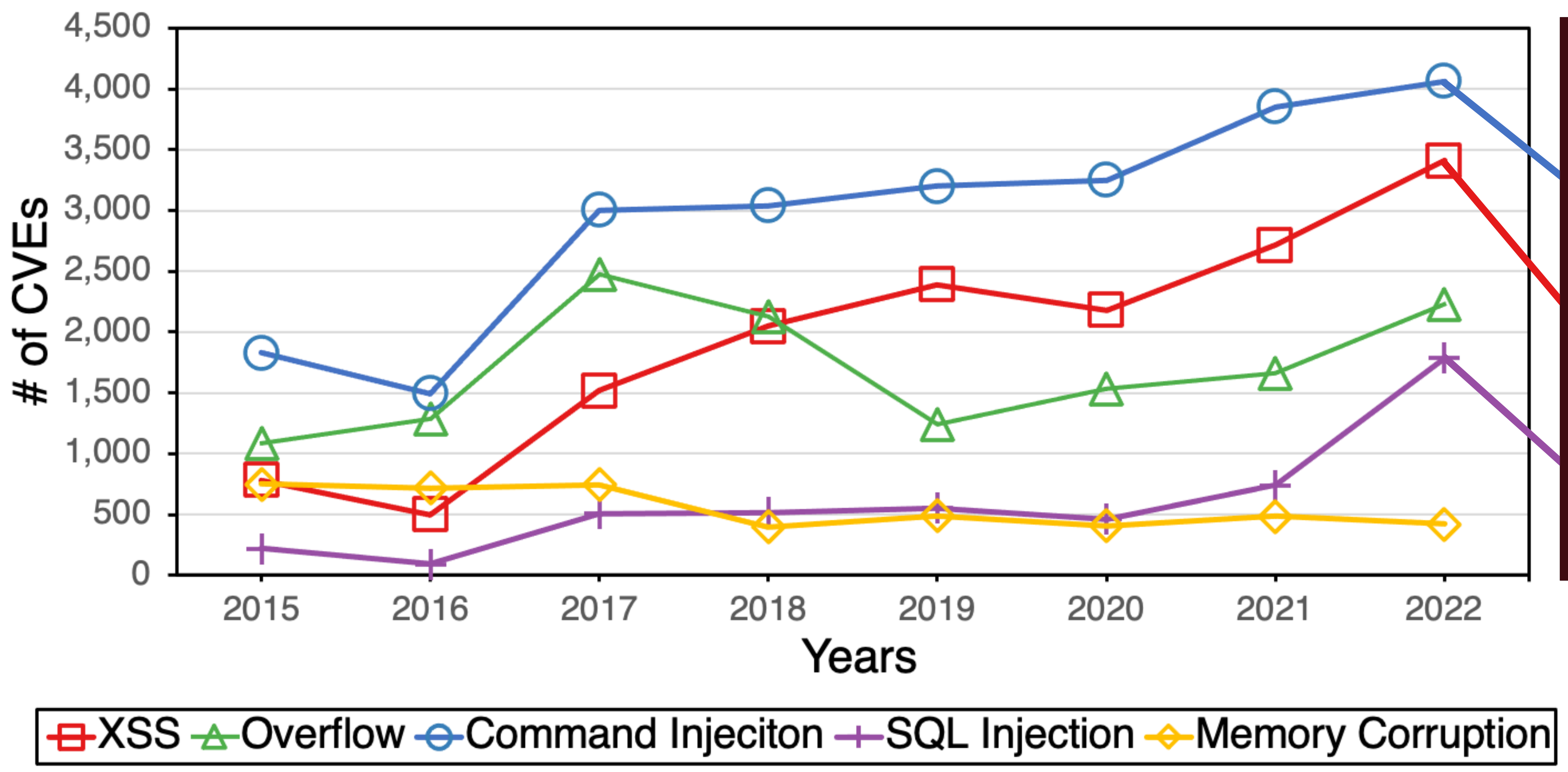
OpenAI ChatGPT

AI models

WebVR

Why Web Security?

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

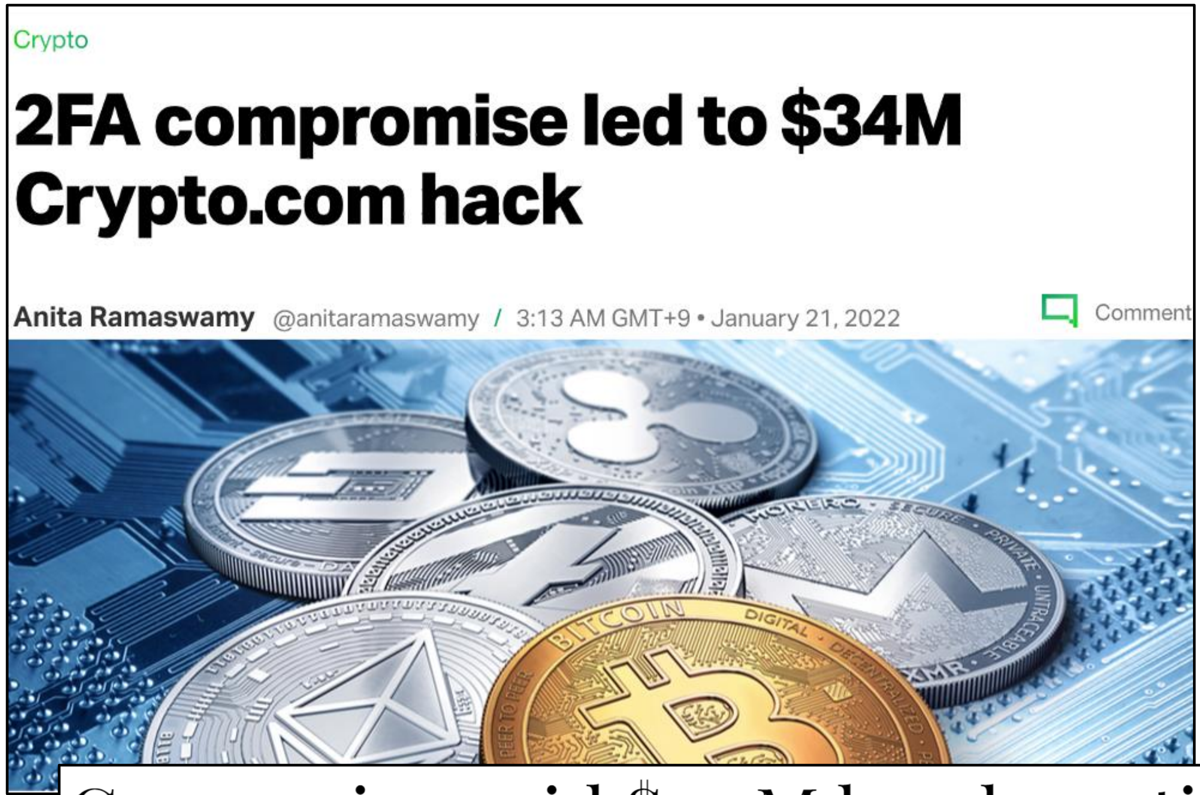


Web attacks

- 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

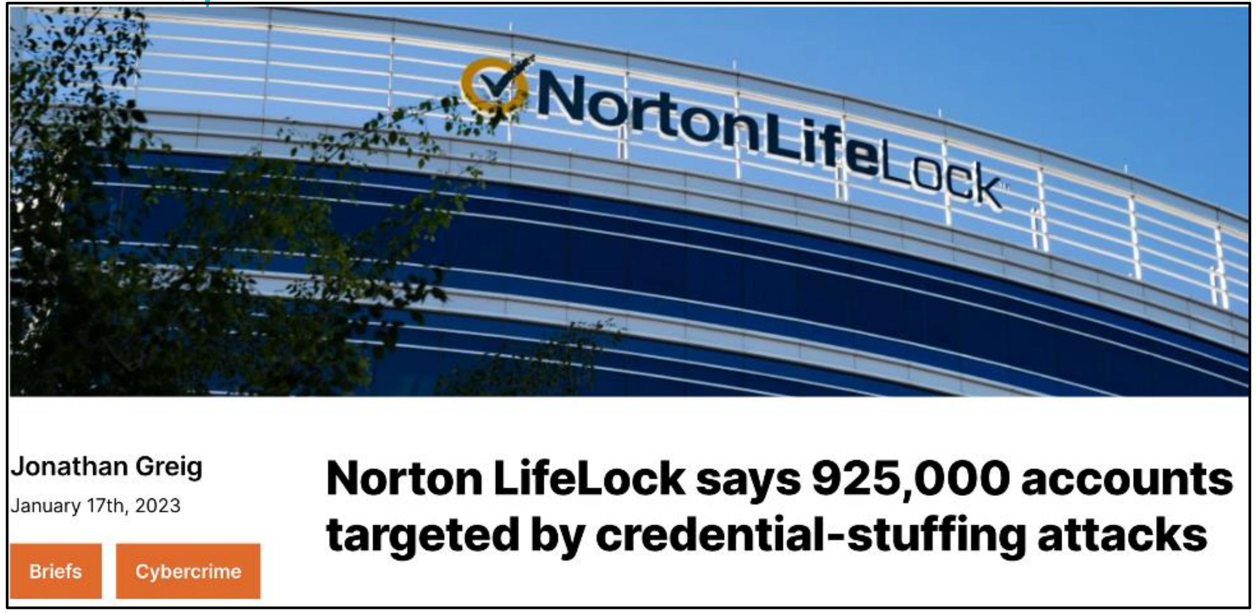


Crypto

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

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

Comment

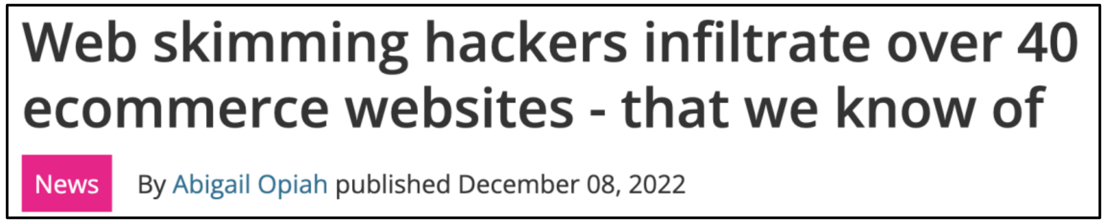


Jonathan Greig

January 17th, 2023

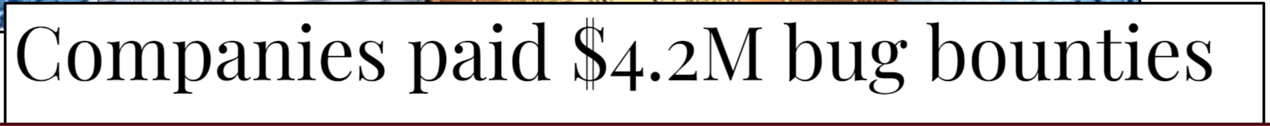
Briefs Cybercrime

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



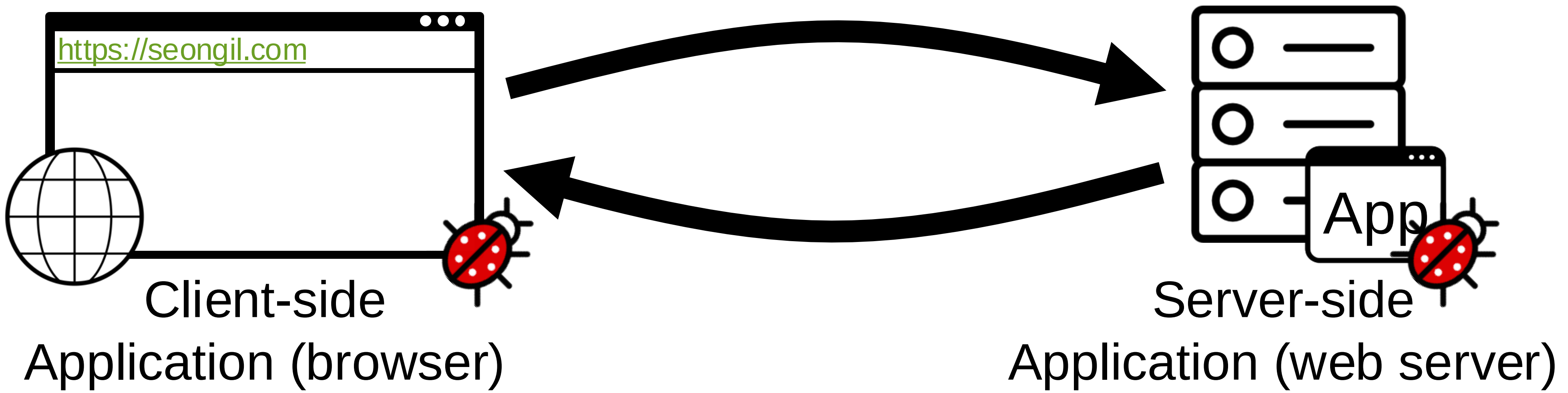
Companies paid $4.2M bug bounties

Web threats are critical!

Introduction to Web Security

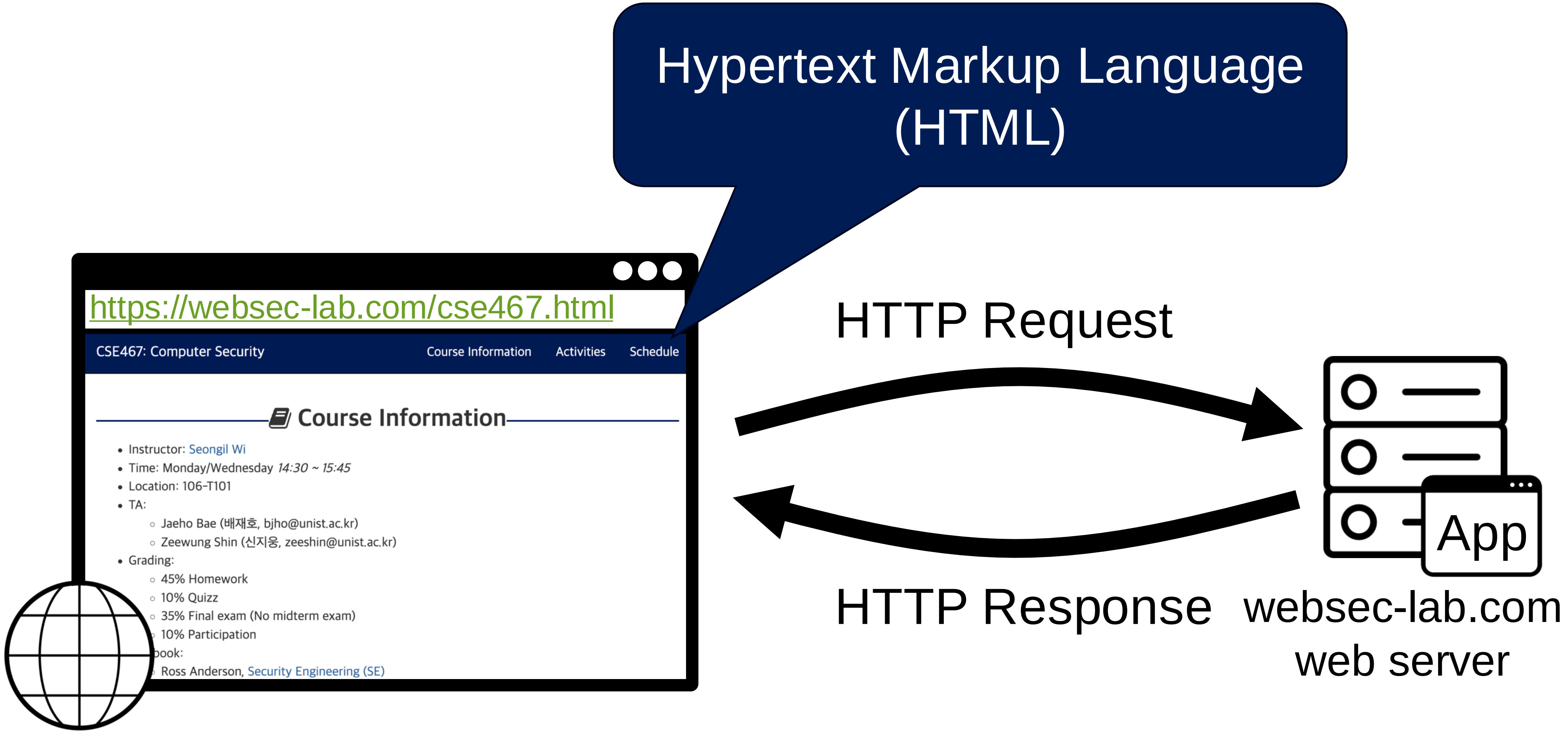
- We are going to study and discuss the web attacks and defenses.

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


https://seongil.com
App
Client-side Application (browser)
Server-side Application (web server)

Web Programming Basic

Web Infrastructure



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

CSE467: Computer Security　Course Information　Activities　Schedule

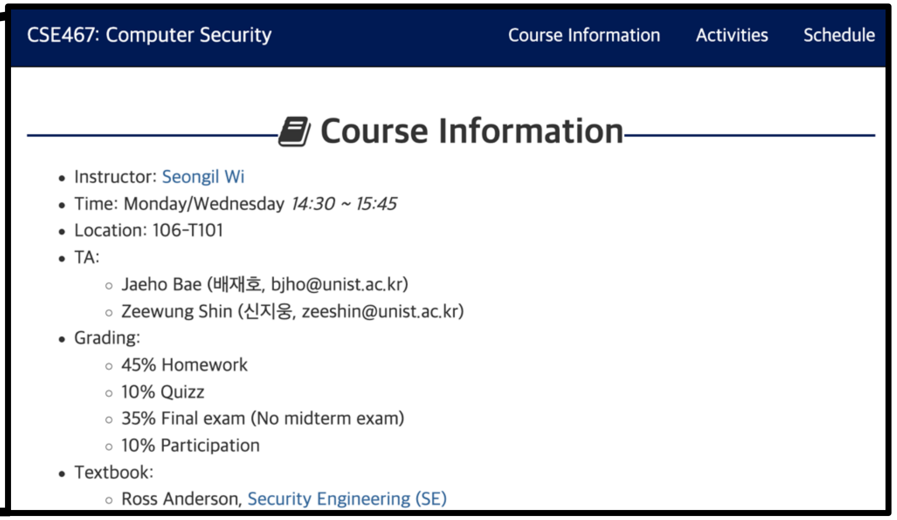
Course Information

- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- ook:
  - Ross Anderson, Security Engineering (SE)

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>
```


CSE467: Computer Security   Course Information   Activities   Schedule

Course Information

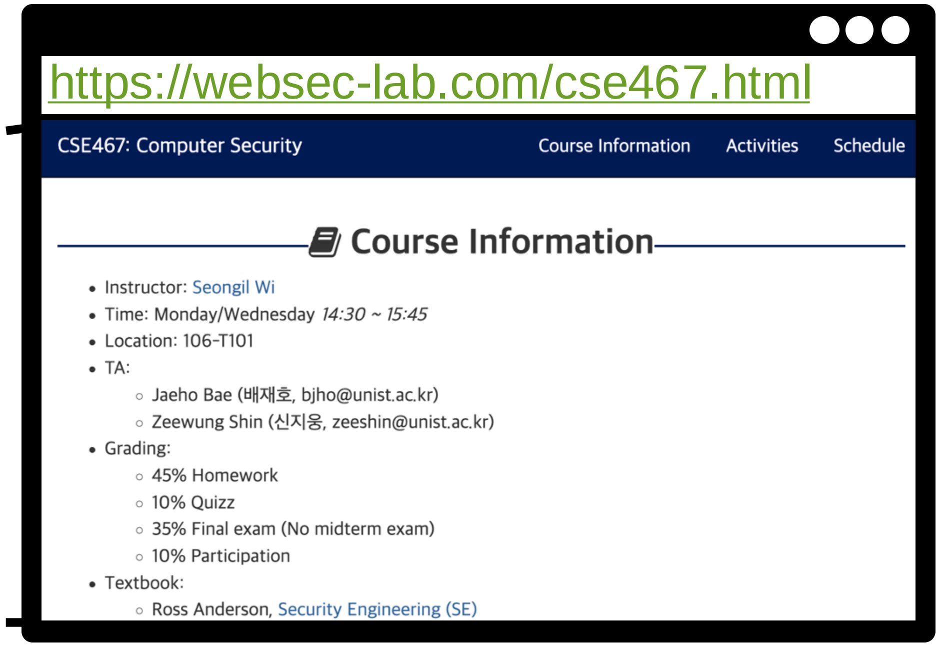
- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - Ross Anderson, Security Engineering (SE)

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

What if we need computation?
⇒ JavaScript!

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



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

CSE467: Computer Security   Course Information   Activities   Schedule

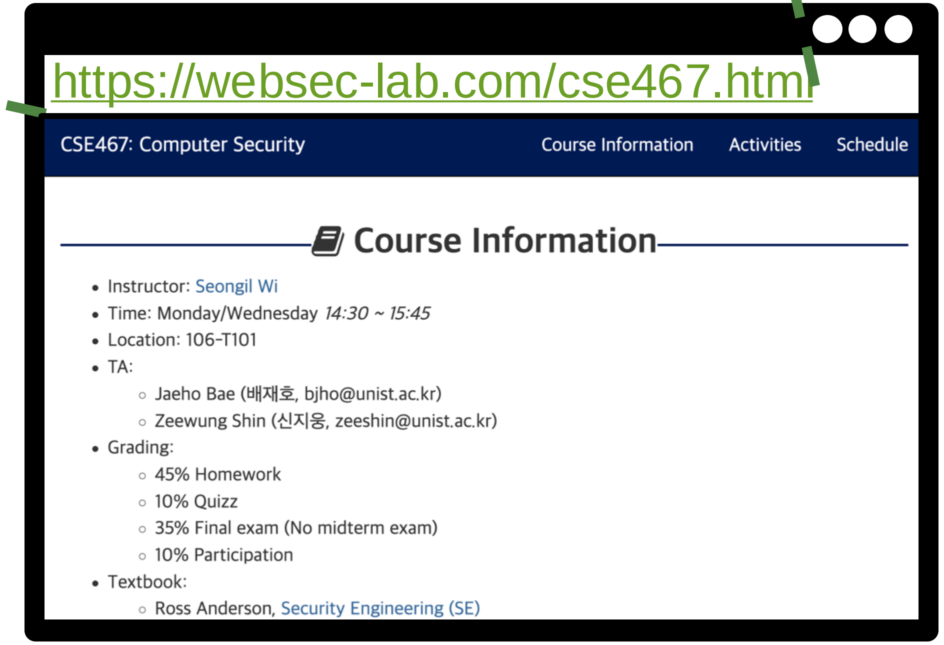
Course Information

- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - Ross Anderson, Security Engineering (SE)

Uniform Resource Locators (URLs)

- Global identifiers of network-retrievable documents
- Example

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



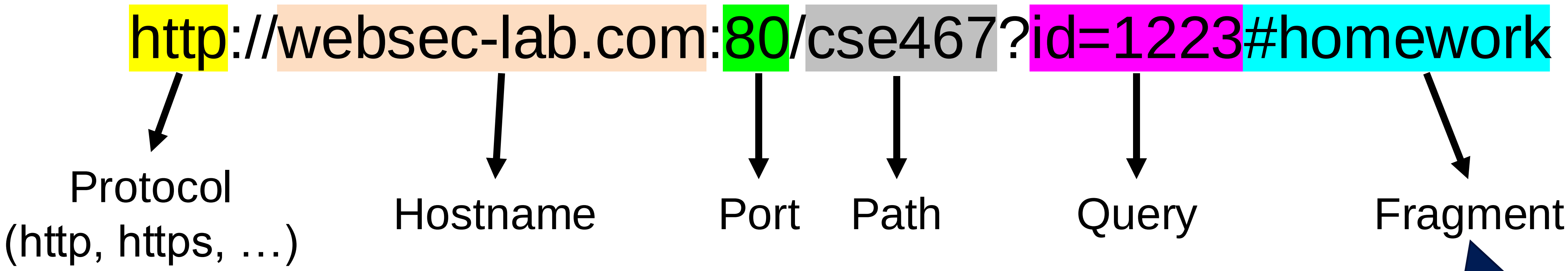
https://websec-lab.com/cse467.html

Course Information

- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - Ross Anderson, Security Engineering (SE)

Uniform Resource Locators (URLs)

- Global identifiers of network-retrievable documents
- Example

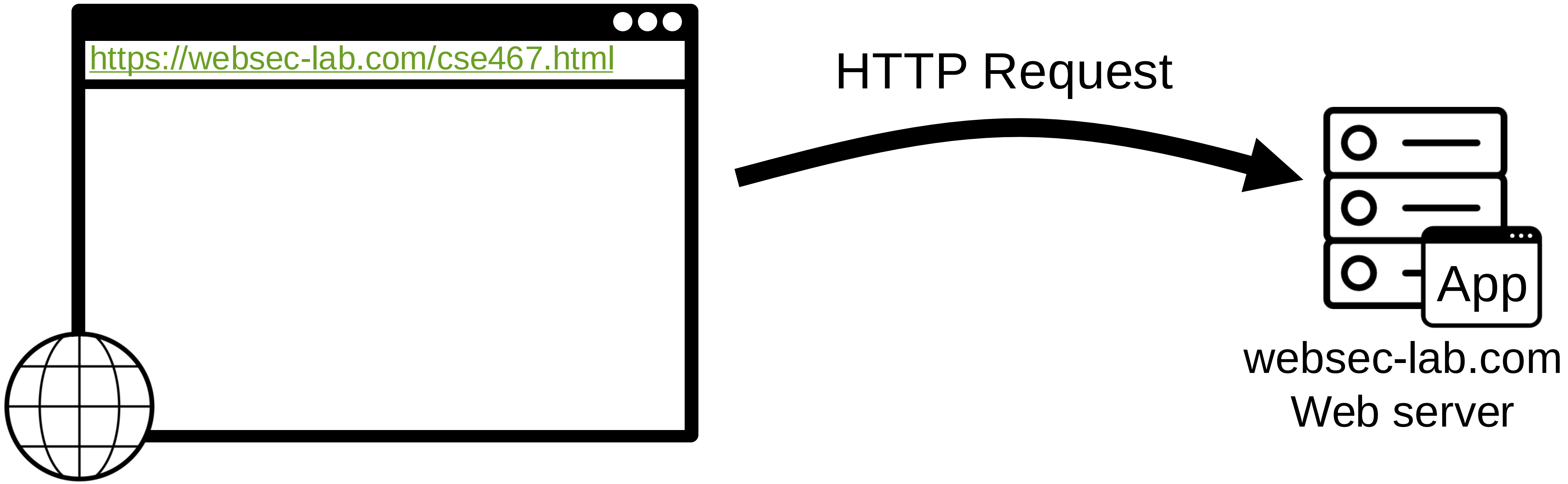

http://websec-lab.com:80/cse467?id=1223#homework
Protocol (http, https, …)
Hostname
Port
Path
Query
Fragment

Fragments are not sent to the server

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

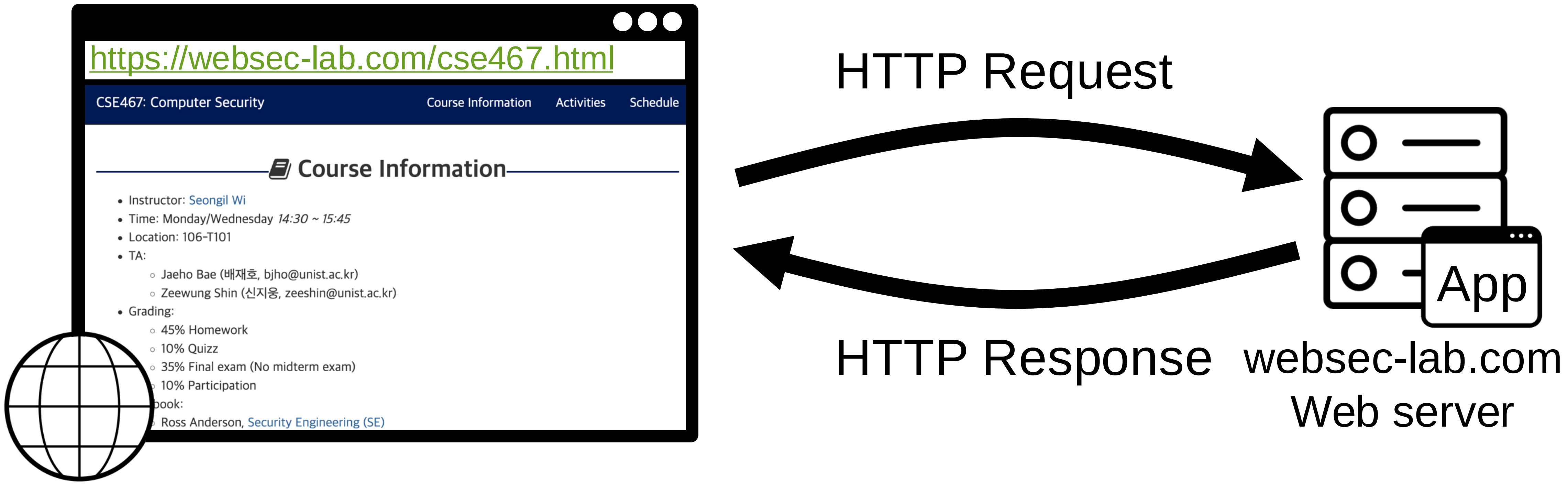
Hyper Text Transfer Protocol (HTTP)

- 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



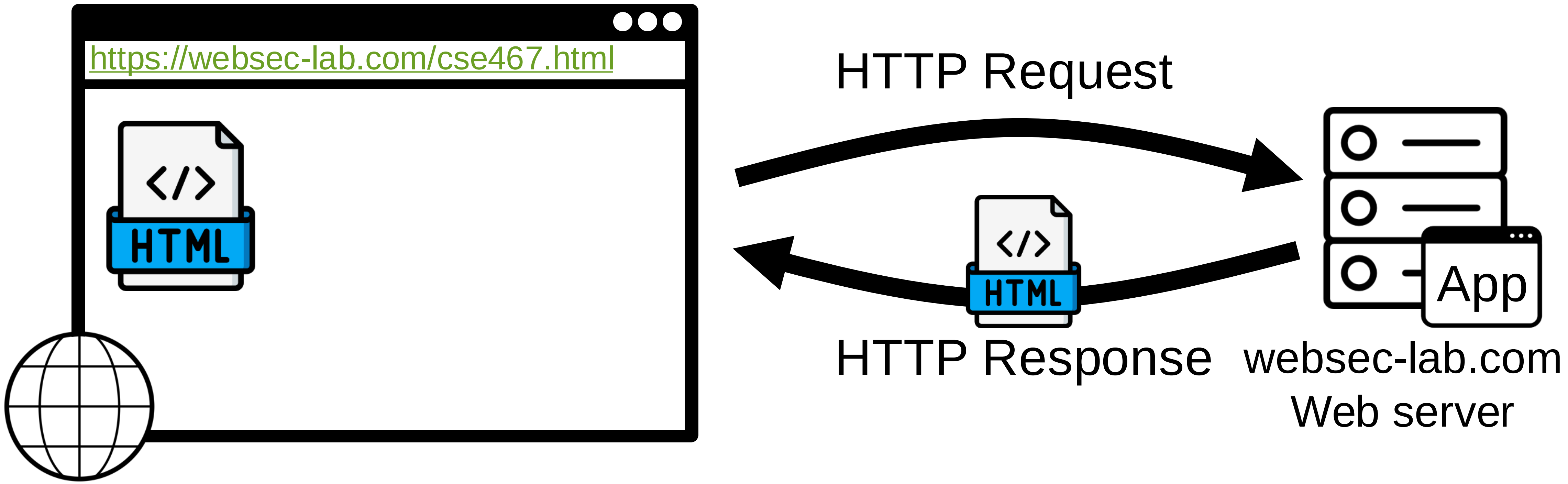
CSE467: Computer Security   Course Information   Activities   Schedule

Course Information

- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- ook:
  - Ross Anderson, Security Engineering (SE)

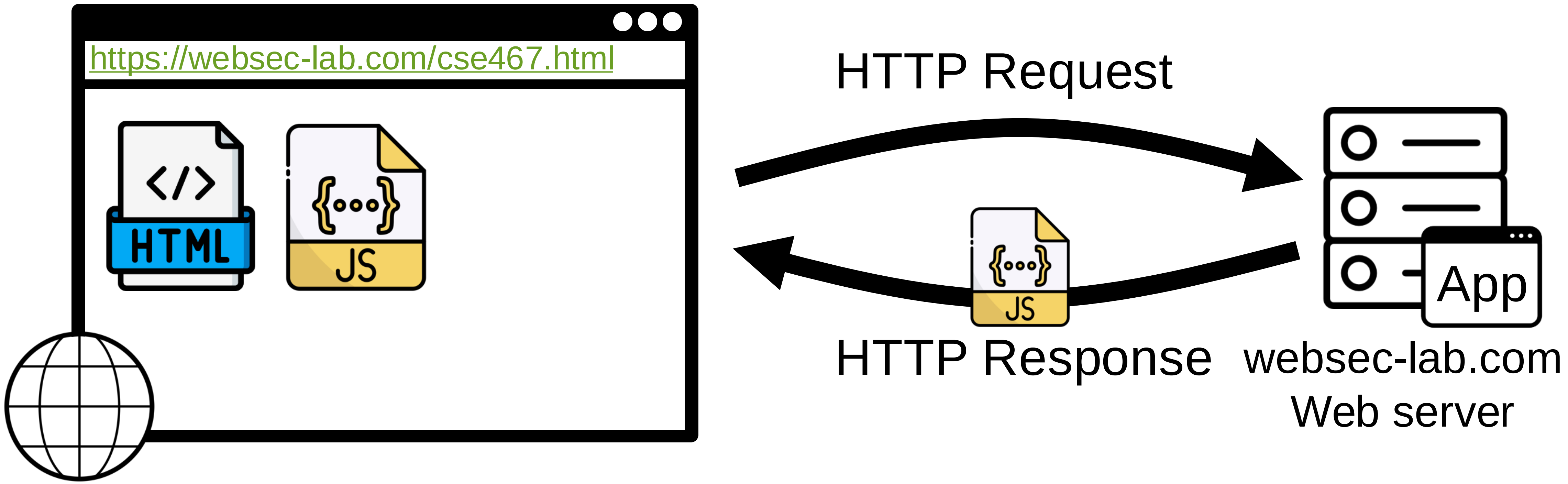
Hyper Text Transfer Protocol (HTTP)

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



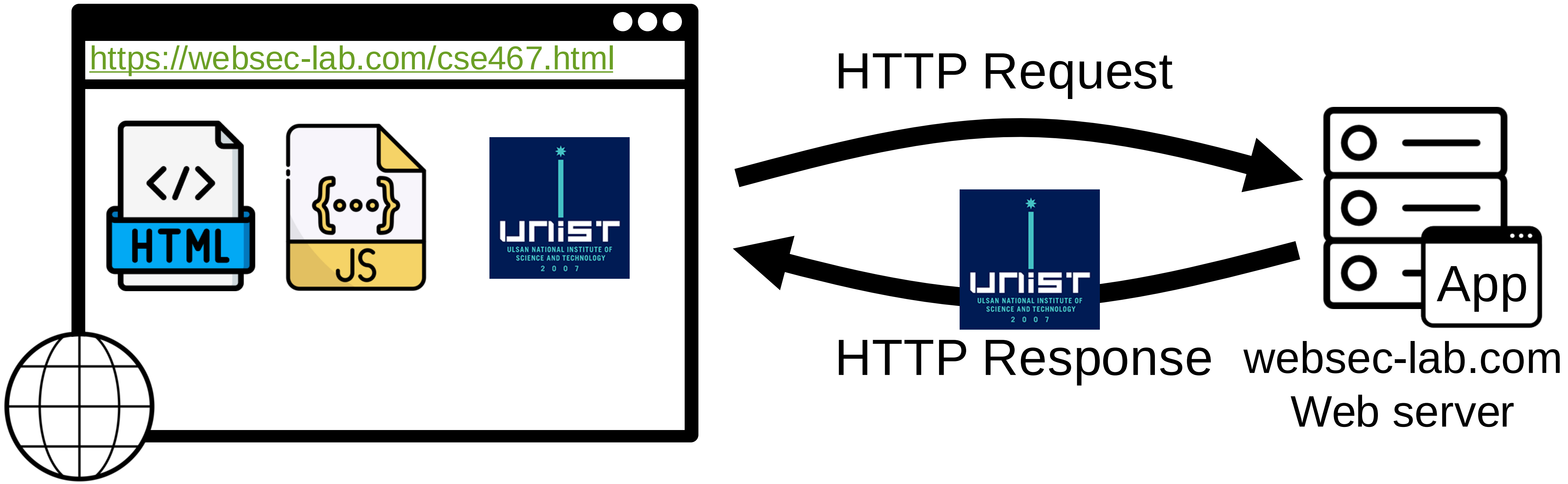
Hyper Text Transfer Protocol (HTTP)

- 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


https://websec-lab.com/cse467.html
HTML
JS
UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY 2007
HTTP Request
HTTP Response
App
websec-lab.com
Web server

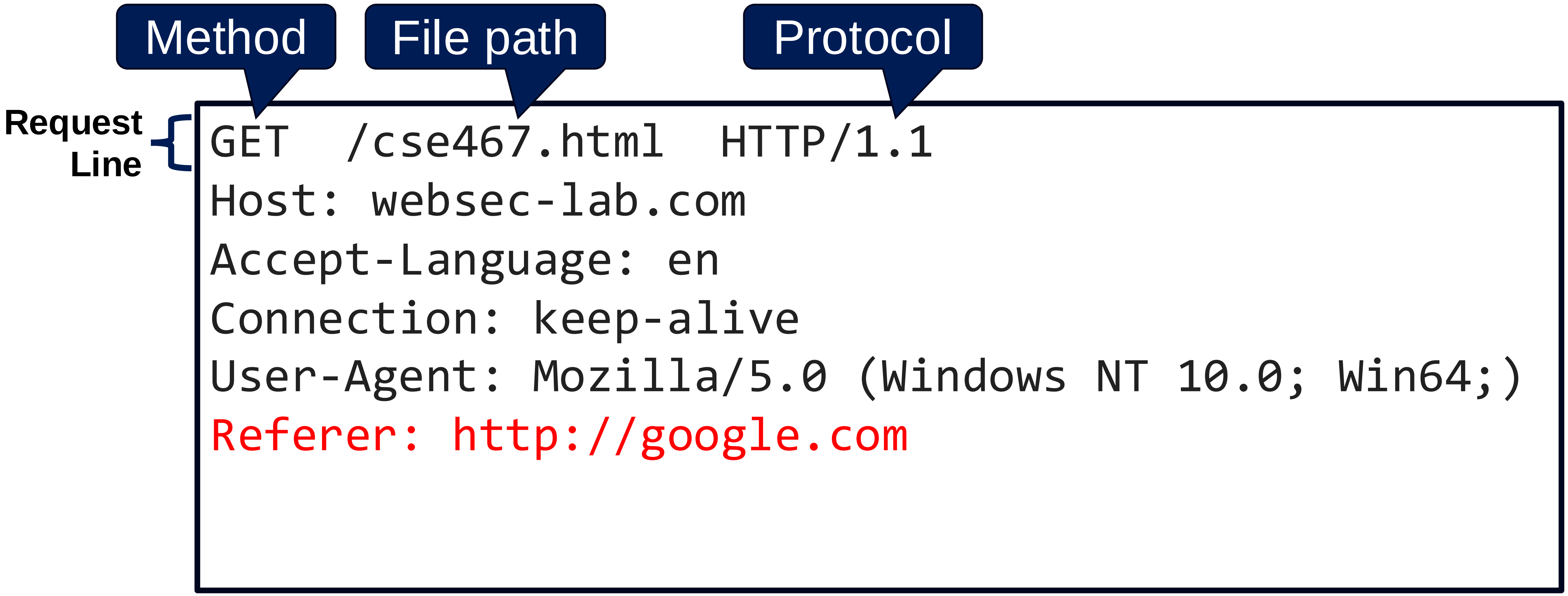
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers
- Application layer protocol
  - A request is sent over a TCP connection on port:80
- Stateless request/response protocol
  - Each request is independent to previous requests

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

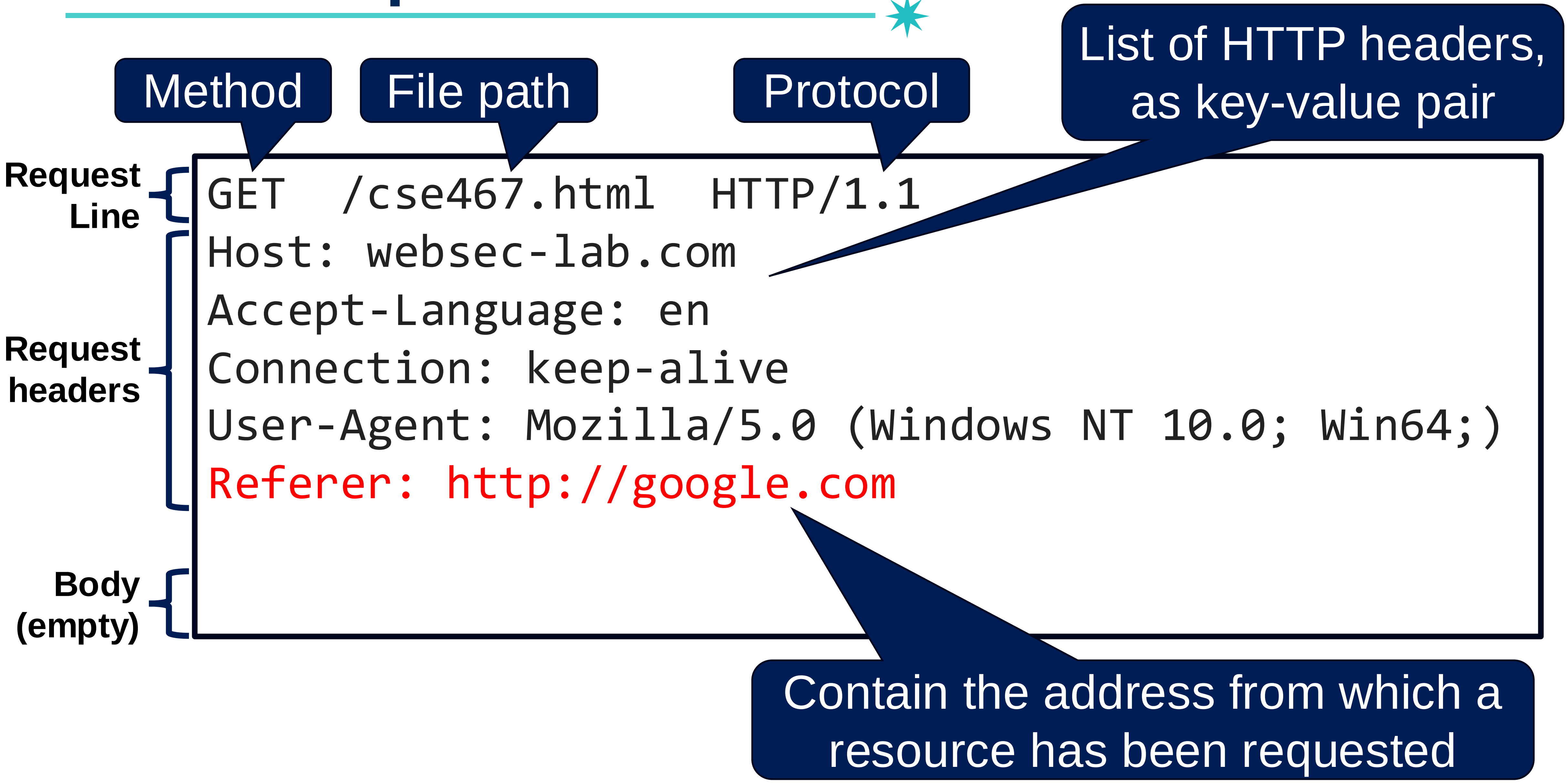


```
GET  /cse467.html  HTTP/1.1
Host: websec-lab.com
Accept-Language: en
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;)
Referer: http://google.com
```

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

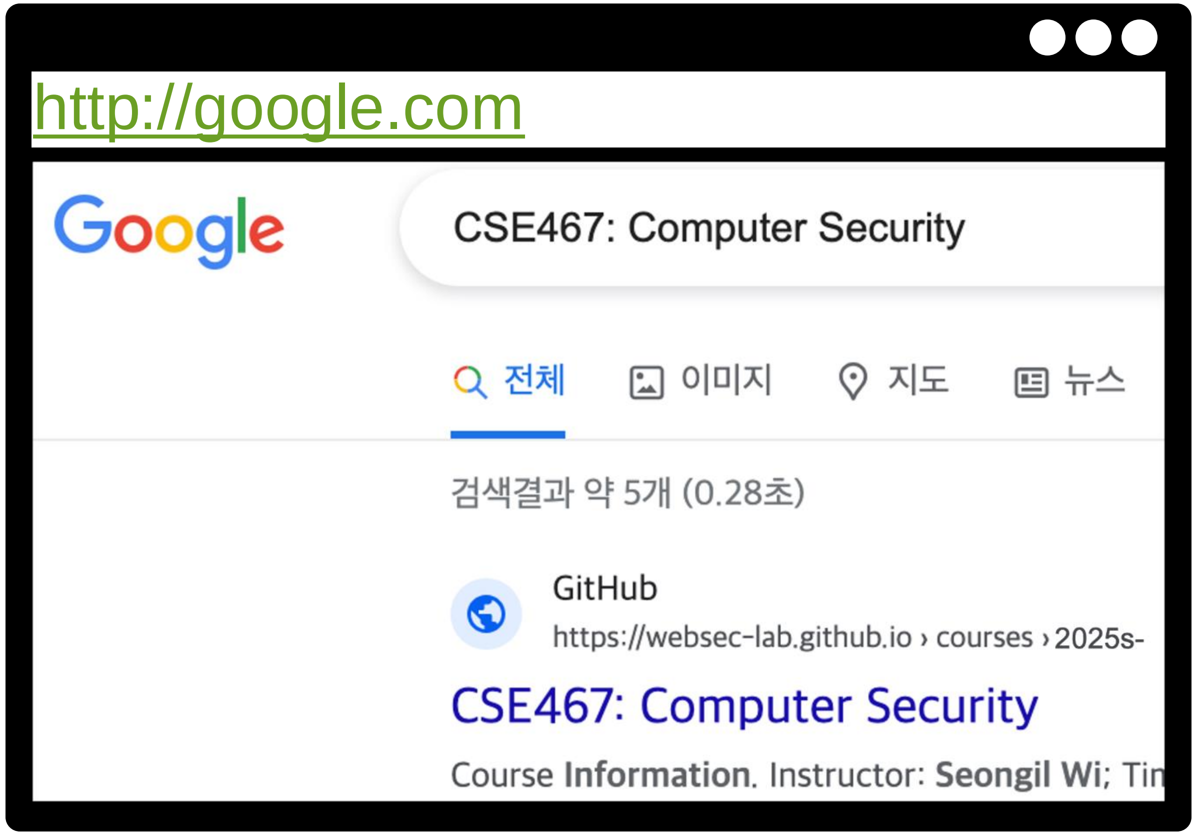
HTTP Request



```
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
```

Method
File path
Protocol
List of HTTP headers, as key-value pair
Request Line
Request headers
Body (empty)
Contain the address from which a resource has been requested

Referrer Header



http://google.com

CSE467: Computer Security

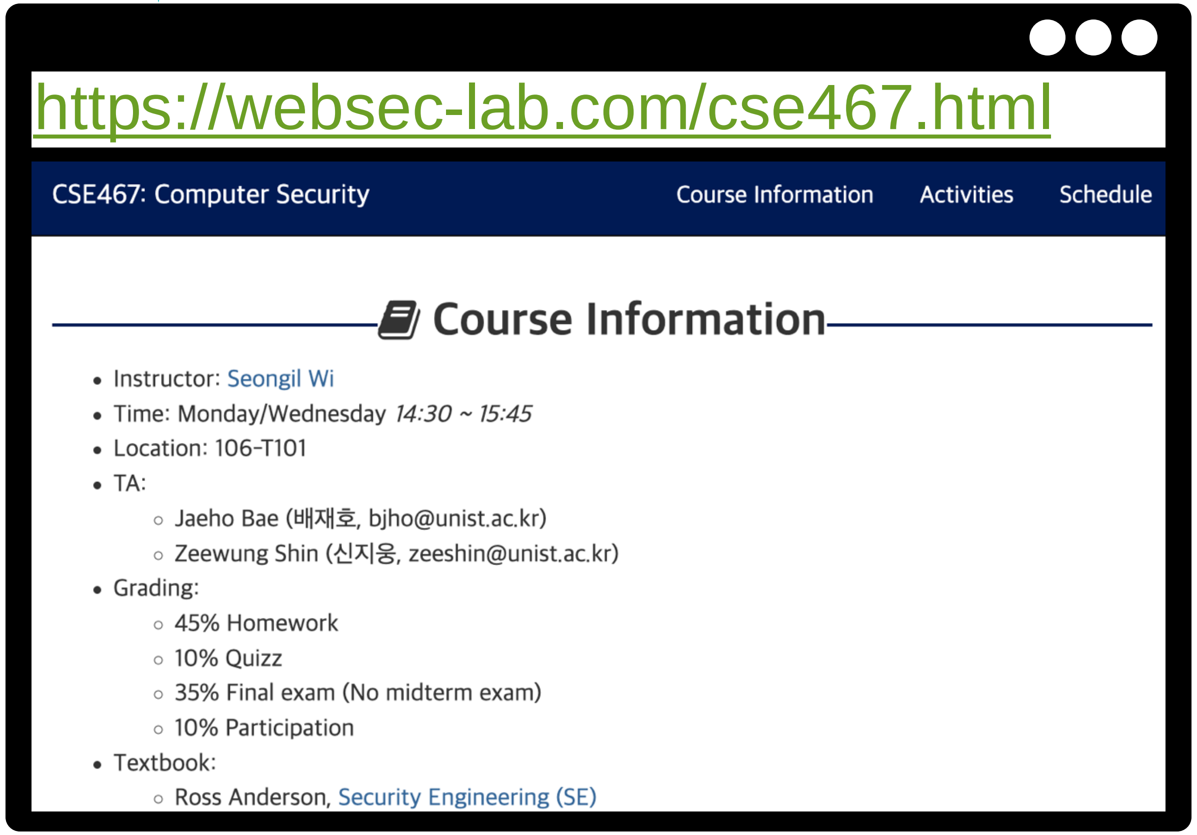
전체 이미지 지도 뉴스

검색결과 약 5개 (0.28초)

GitHub
https://websec-lab.github.io › courses › 2025s-

CSE467: Computer Security

Course **Information**. Instructor: **Seongil Wi**; Tin



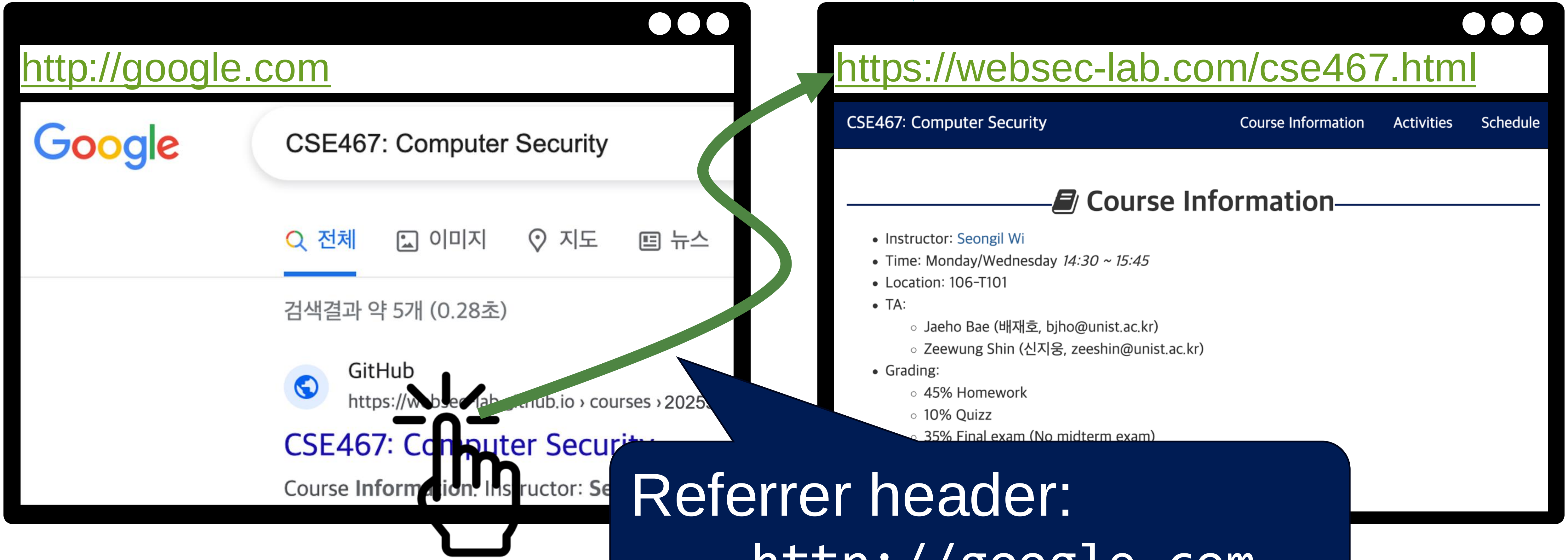
https://websec-lab.com/cse467.html

CSE467: Computer Security — Course Information — Activities — Schedule

Course Information

- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - Ross Anderson, Security Engineering (SE)

Referrer Header

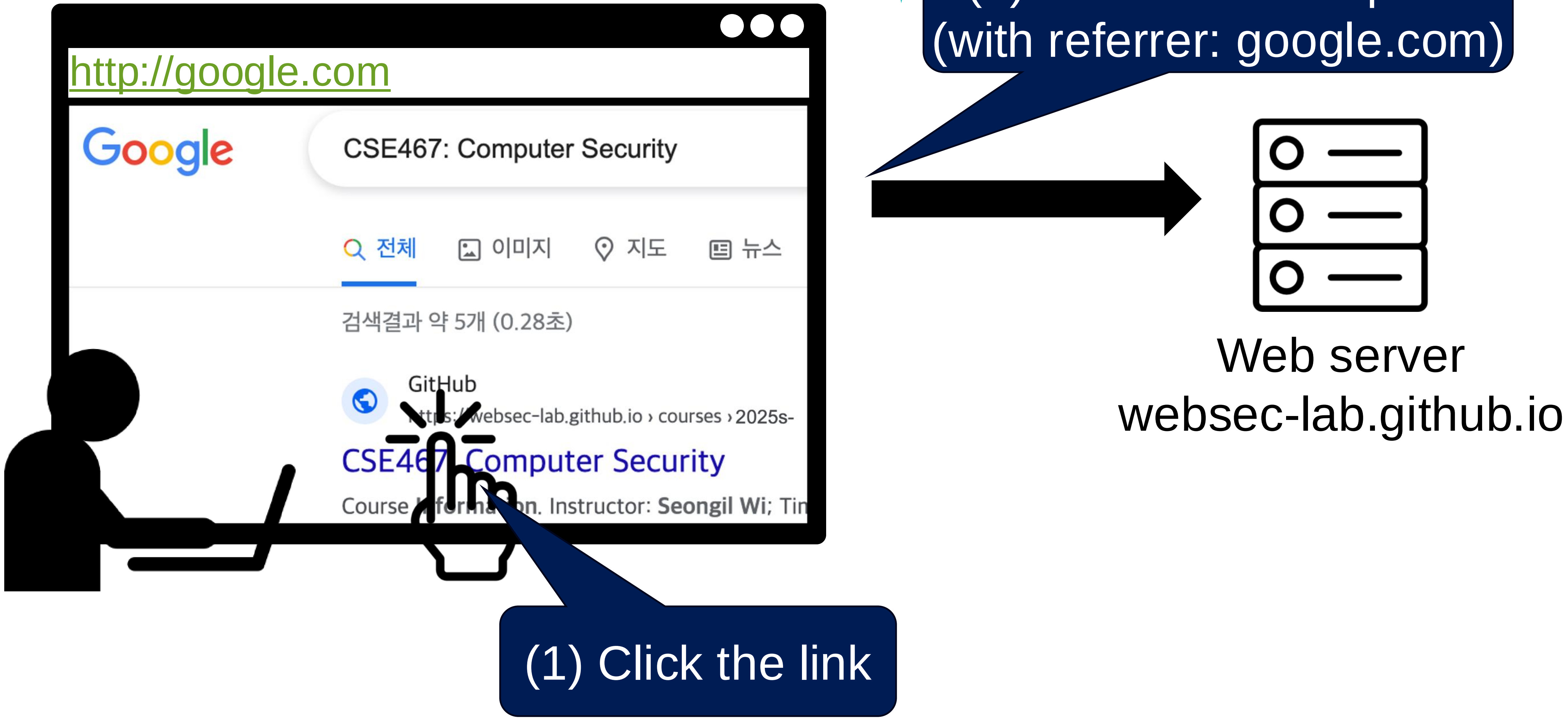

http://google.com
Google
CSE467: Computer Security
전체
이미지
지도
뉴스
검색결과 약 5개 (0.28초)
GitHub
https://websec-lab.github.io › courses › 2025
CSE467: Computer Security
Course Information. Instructor: Se
https://websec-lab.com/cse467.html
CSE467: Computer Security
Course Information
Activities
Schedule

Course Information

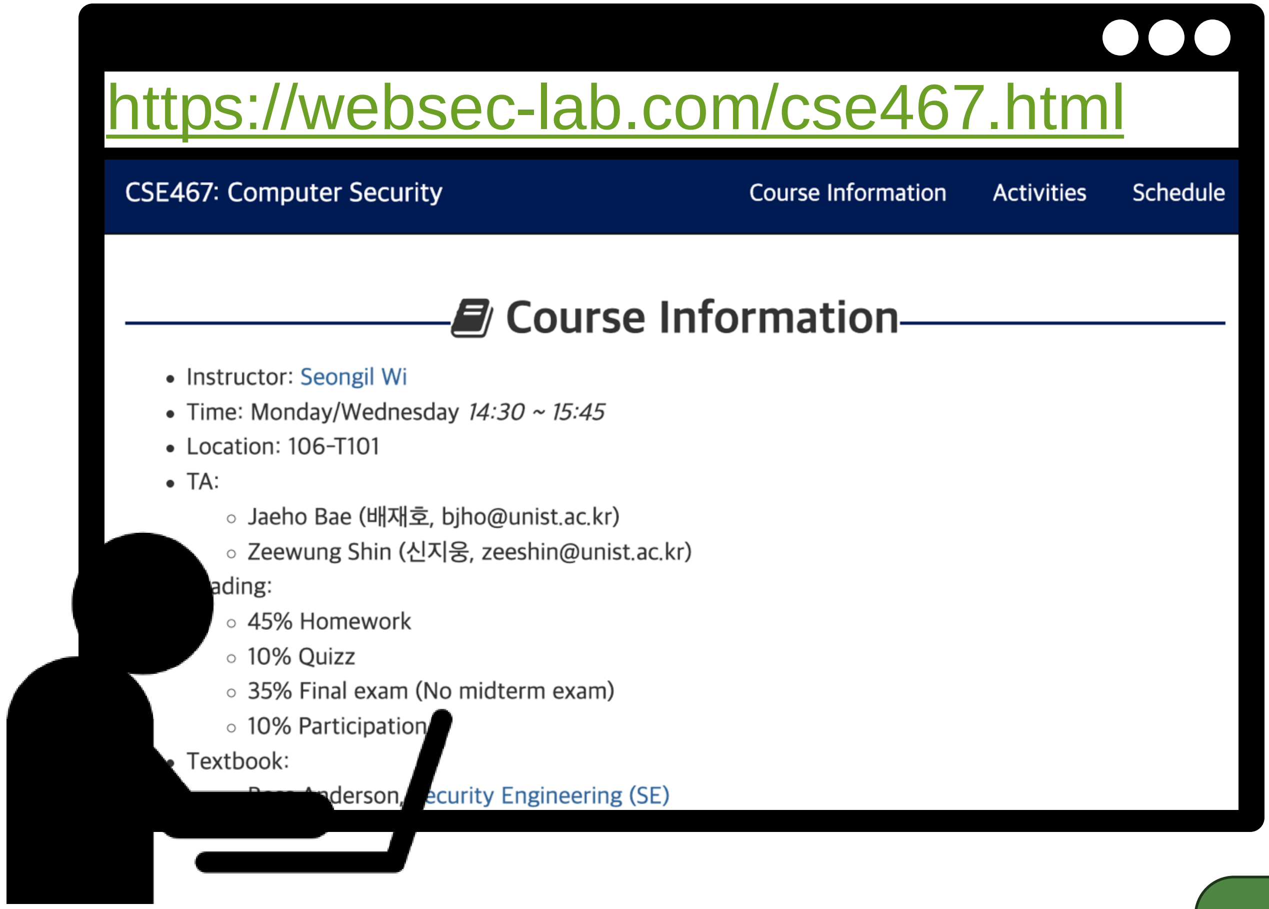
- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- Grading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)

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

Referrer Header in Detail


http://google.com
Google
CSE467: Computer Security
전체
이미지
지도
뉴스
검색결과 약 5개 (0.28초)
GitHub
https://websec-lab.github.io › courses › 2025s-
CSE467: Computer Security
Course Information. Instructor: Seongil Wi; Tin
(1) Click the link
(2) Send HTTP request (with referrer: google.com)
Web server
websec-lab.github.io

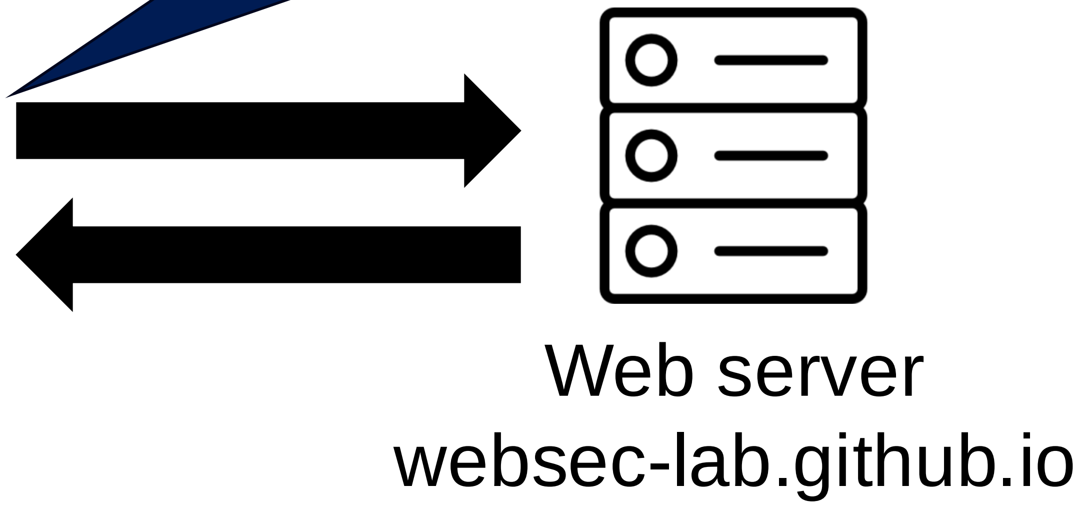
Referrer Header in Detail


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

CSE467: Computer Security | Course Information | Activities | Schedule

Course Information

- Instructor: Seongil Wi
- Time: Monday/Wednesday *14:30 ~ 15:45*
- Location: 106-T101
- TA:
  - Jaeho Bae (배재호, bjho@unist.ac.kr)
  - Zeewung Shin (신지웅, zeeshin@unist.ac.kr)
- ading:
  - 45% Homework
  - 10% Quizz
  - 35% Final exam (No midterm exam)
  - 10% Participation
- Textbook:
  - derson, ecurity Engineering (SE)

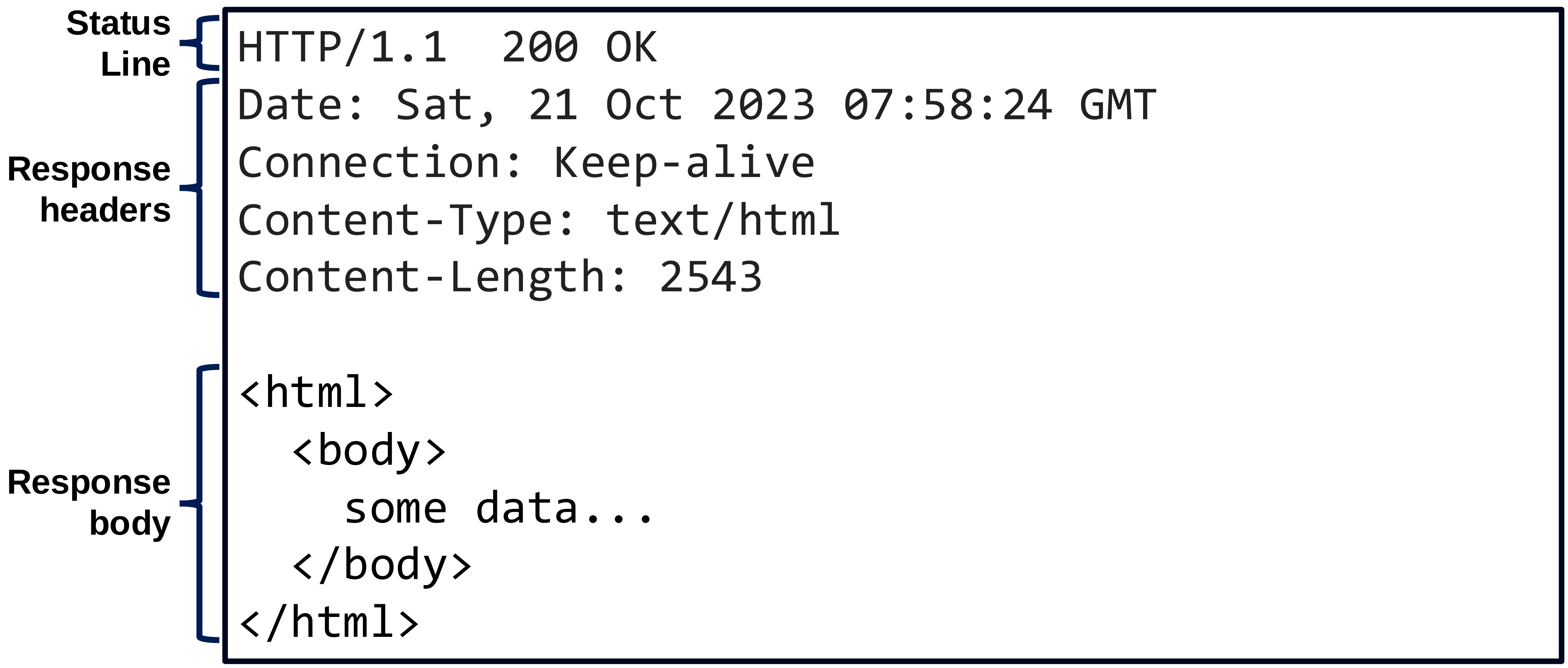


The server can analyze **where the request originated**

Question

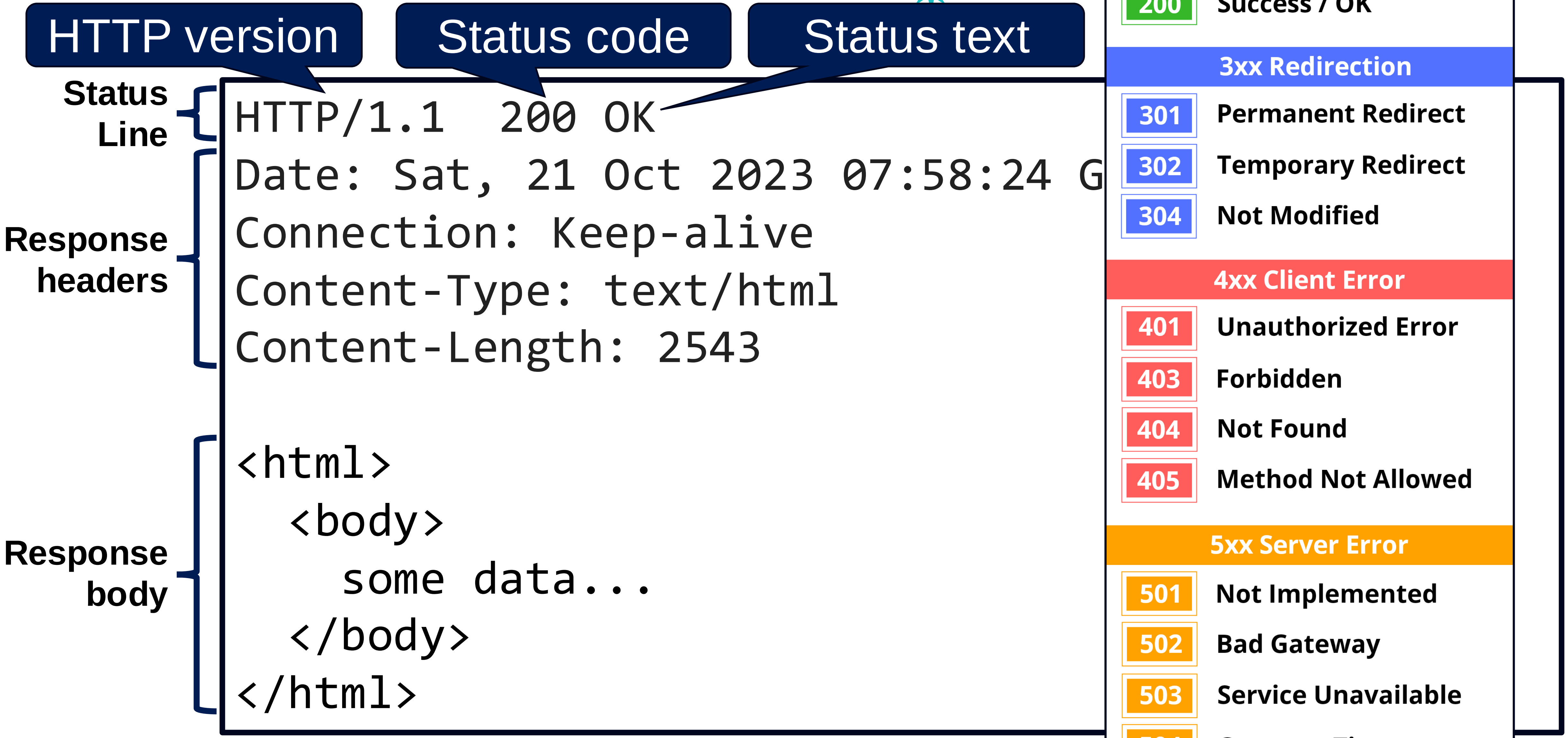
Are there any security issues with the referrer header?

HTTP Response



```
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



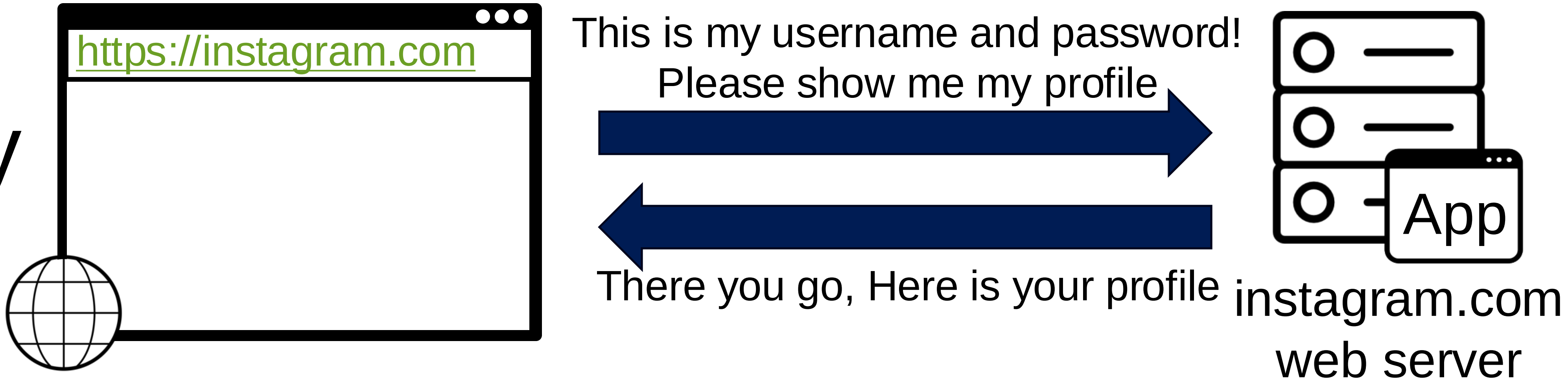
```
HTTP/1.1  200 OK
Date: Sat, 21 Oct 2023 07:58:24 G
Connection: Keep-alive
Content-Type: text/html
Content-Length: 2543

<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

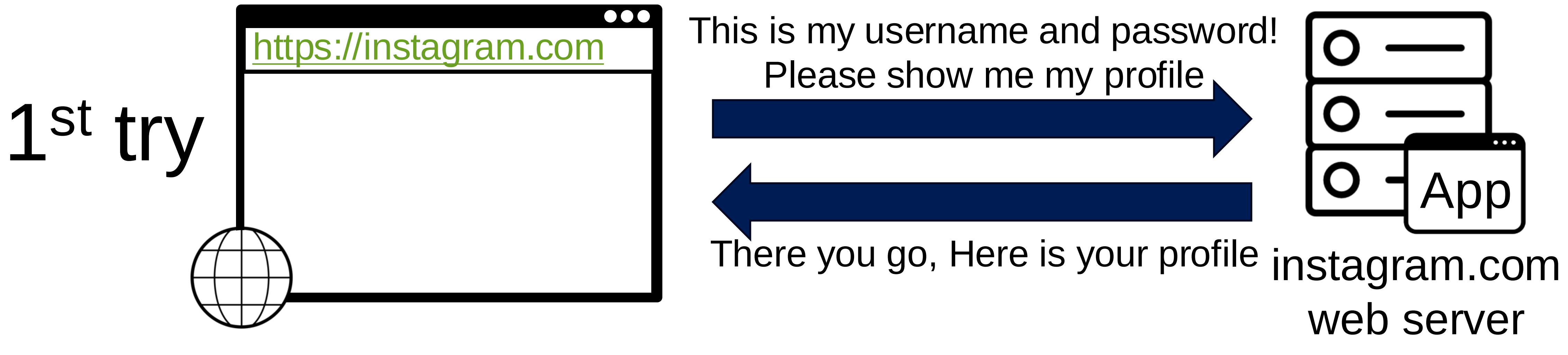
HTTP is a Stateless Protocol

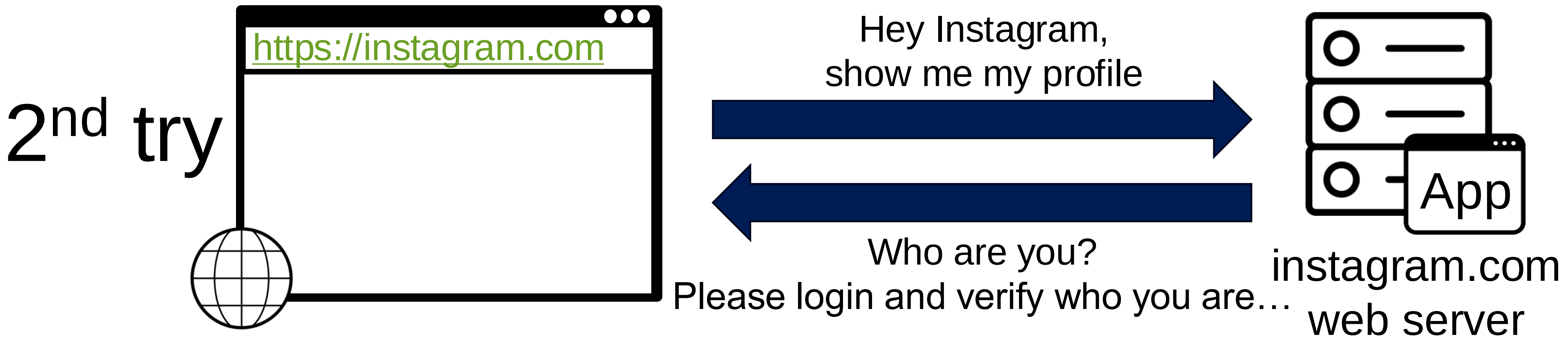
1st try



https://instagram.com
This is my username and password!
Please show me my profile
There you go, Here is your profile
App
instagram.com
web server

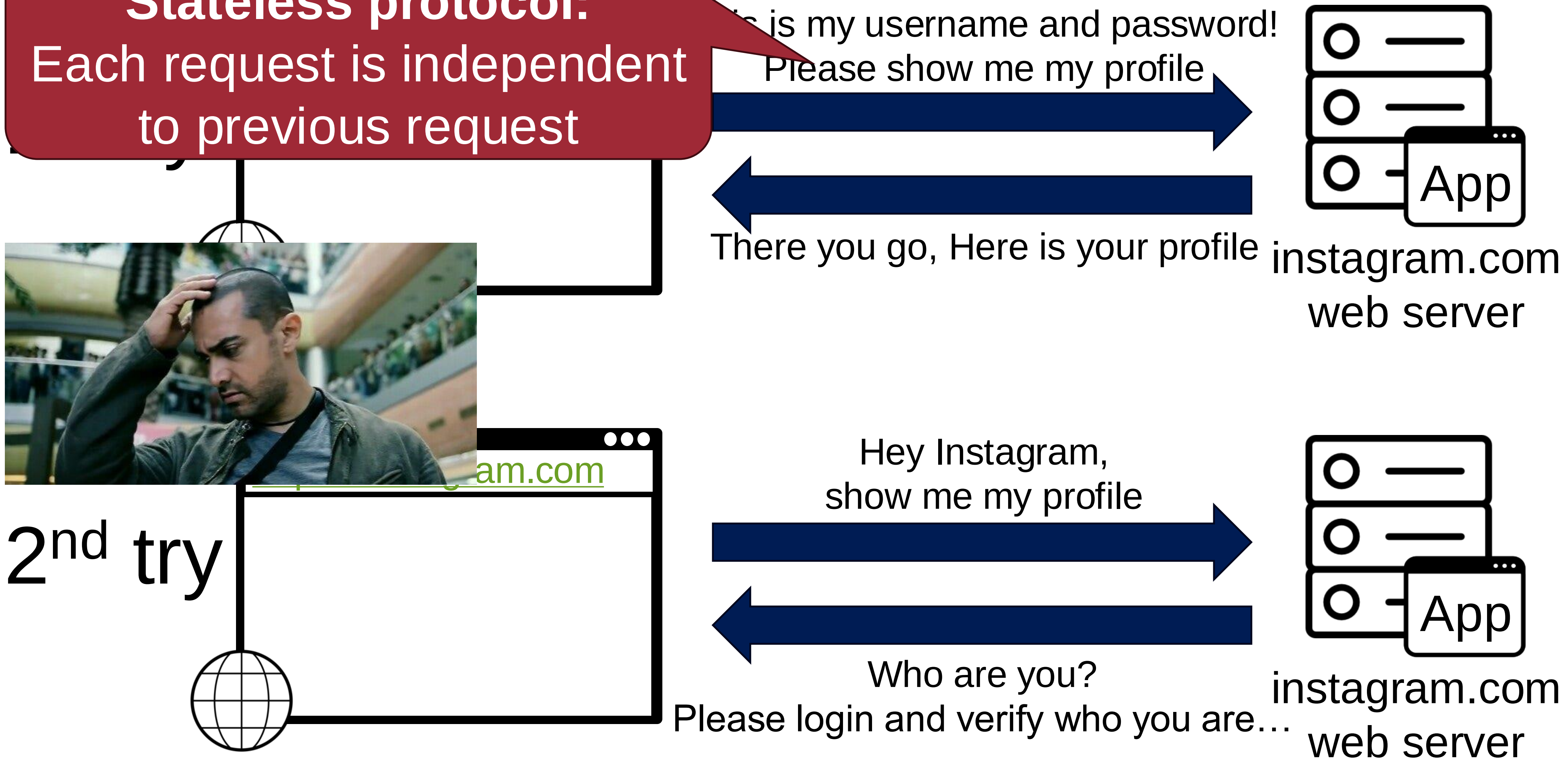
HTTP is a Stateless Protocol


1st try
https://instagram.com
This is my username and password!
Please show me my profile
There you go, Here is your profile
App
instagram.com
web server



HTTP is a Stateless Protocol

Stateless protocol:
Each request is independent to previous request



…is is my username and password! Please show me my profile
There you go, Here is your profile
instagram.com web server
App
…am.com
2nd try
Hey Instagram, show me my profile
Who are you? Please login and verify who you are…
instagram.com web server
App

Question

How to make HTTP “act” stateful?

Adding State to HTTP

- Recall: no inherent state in HTTP
  - Server does not keep any state after the connection is closed

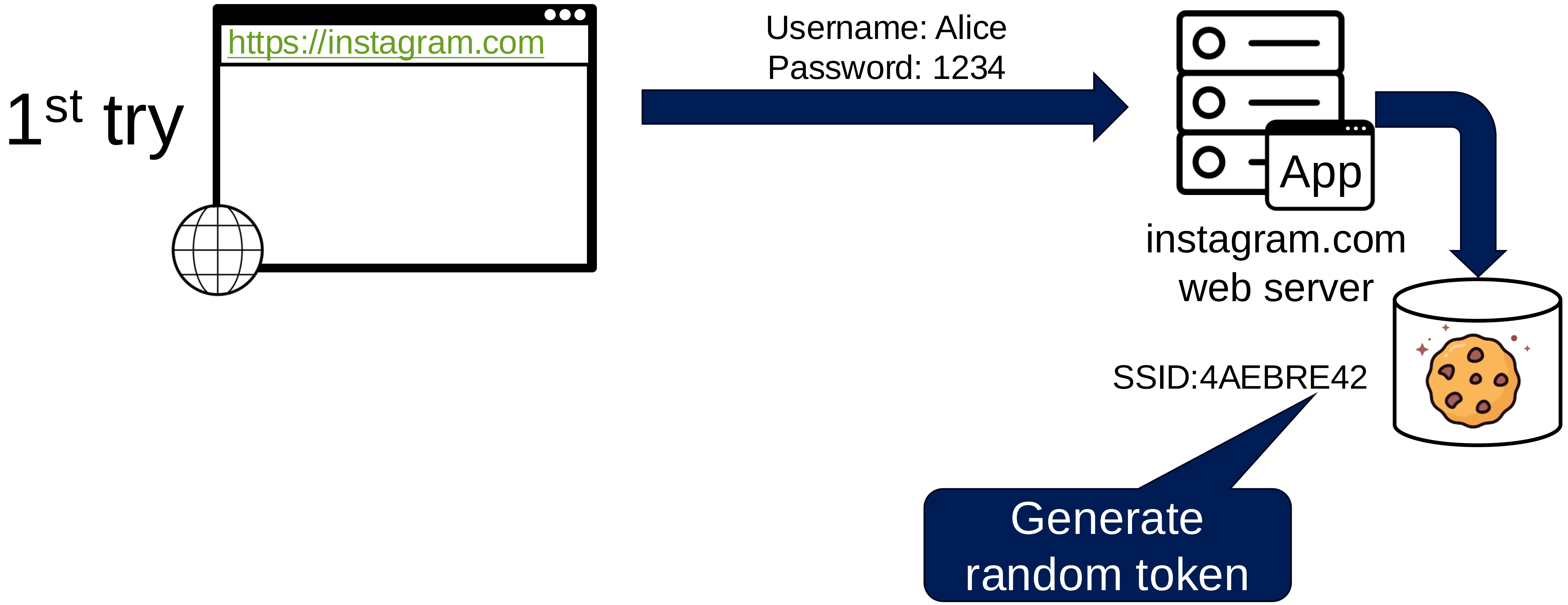
- For static content sites, no problem
  - Developing “applications” is impossible though
  - E.g., shopping cart on Amazon

- Need to introduce state in HTTP
  - in the form of “cookies”

Cookie: Making HTTP Stateful

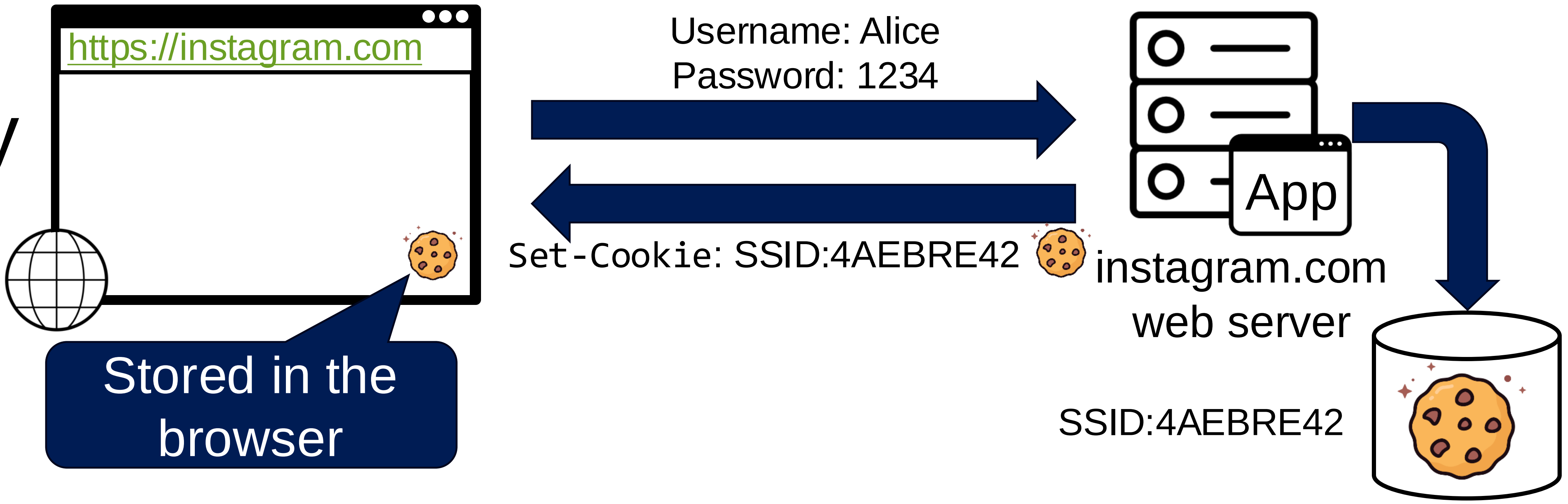
- HTTP cookie: small piece of data that a server sends to the browser, who stores it and sends it back with subsequent requests

Cookie: Making HTTP Stateful

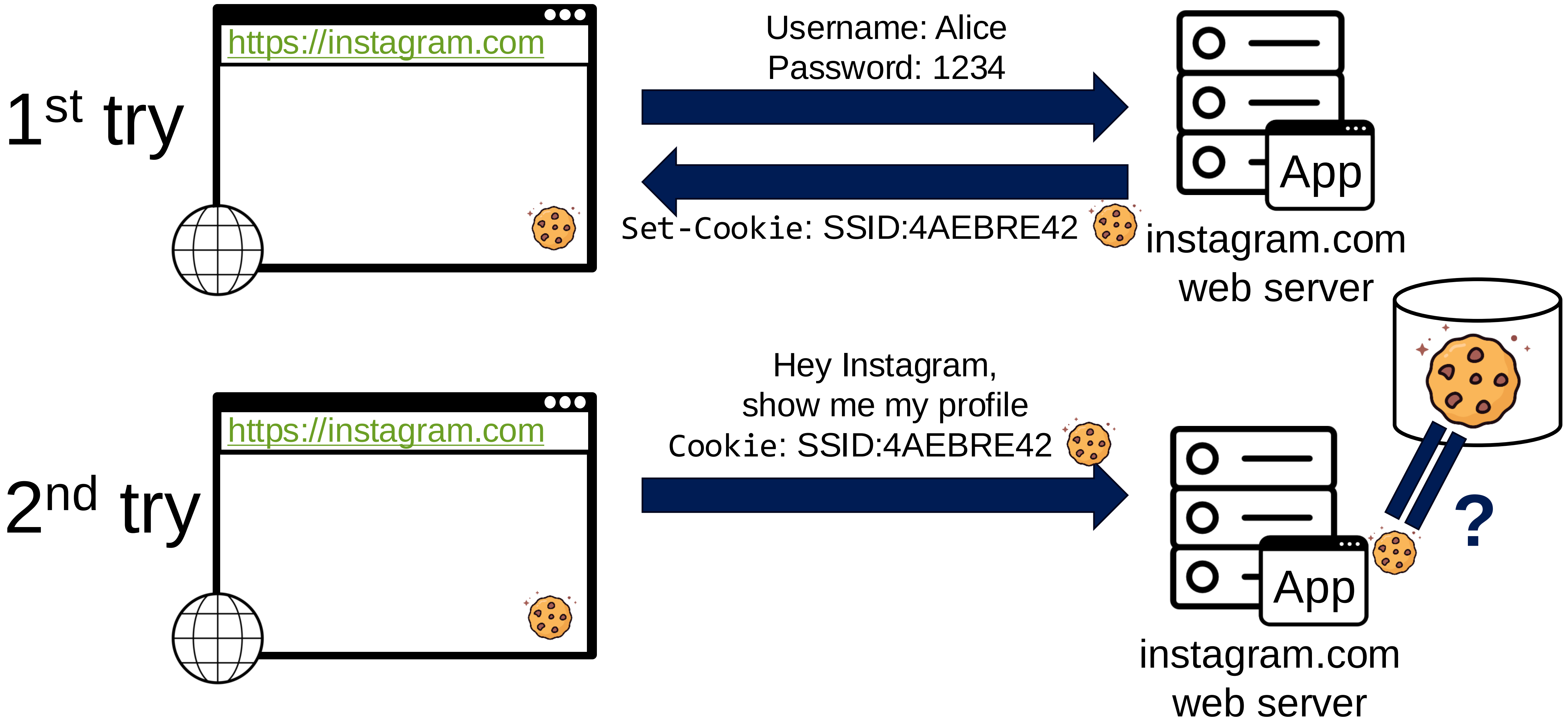


Cookie: Making HTTP Stateful

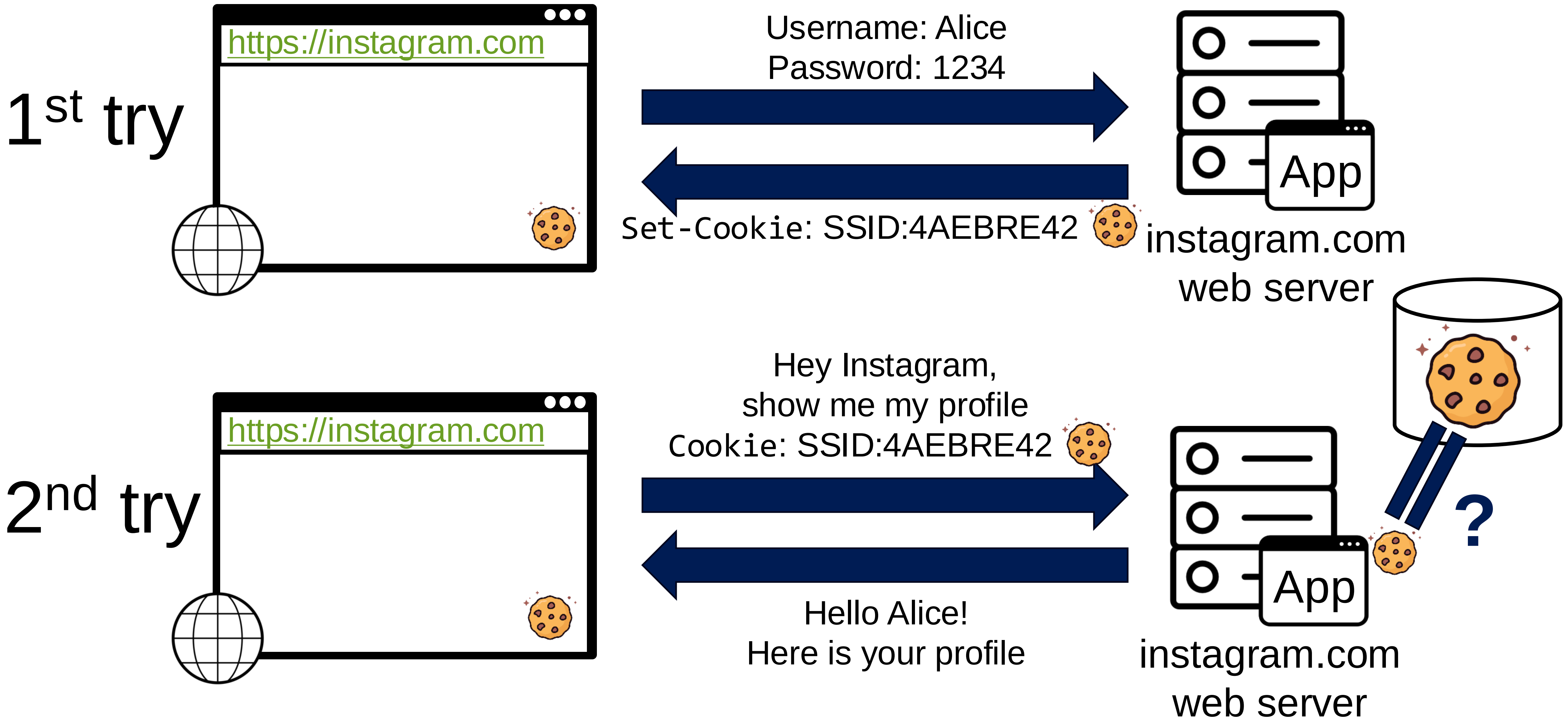
1st try



Cookie: Making HTTP Stateful



Cookie: Making HTTP Stateful



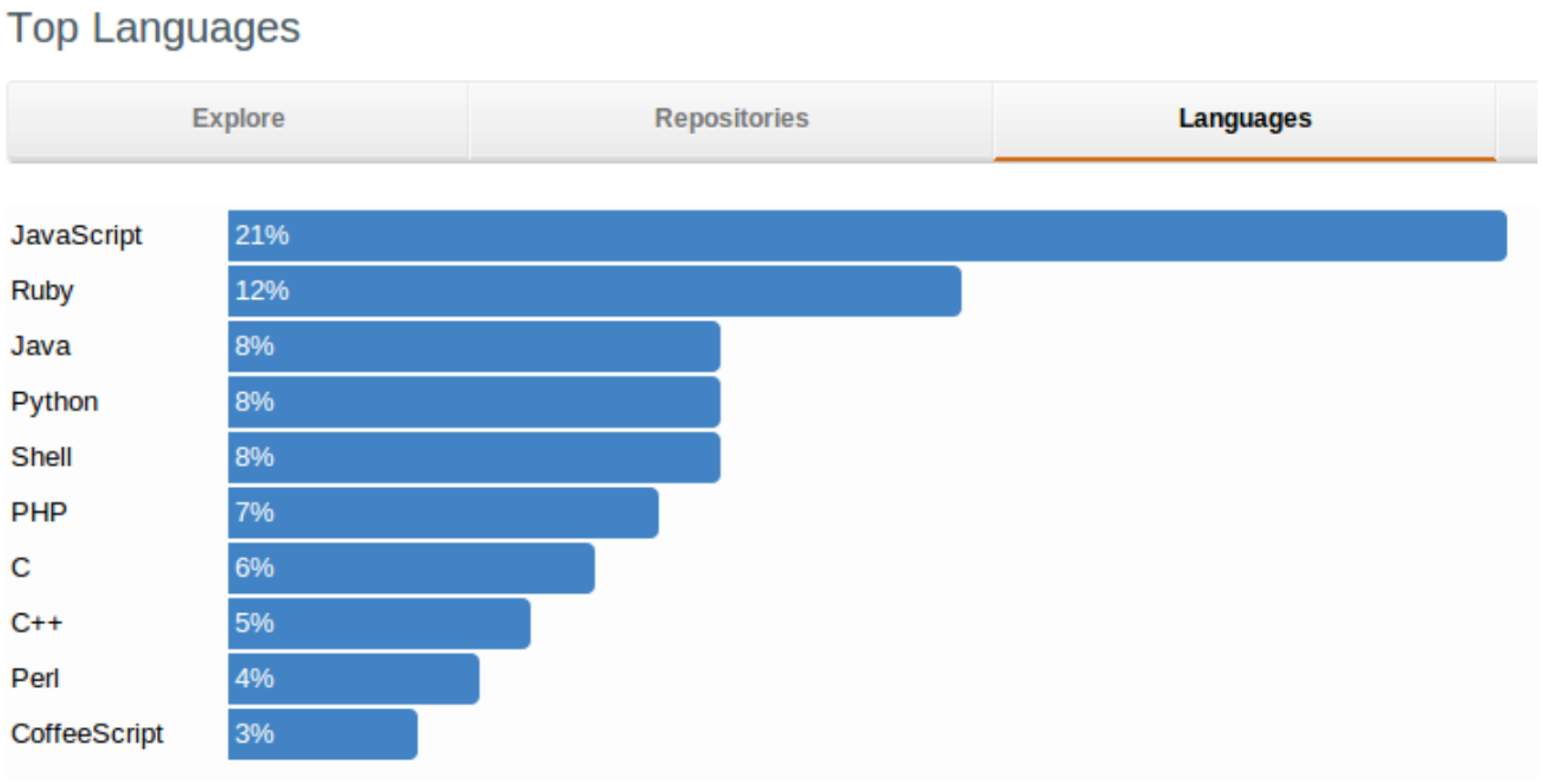
Cookie: Making HTTP Stateful

- Generate random token on first page visit
- Sent to client via `Set-Cookie` header
- Client always sends along cookies in every request to the server

- Cookies are persisted in the browser
  - Controllable by `Expires` option in cookie

JavaScript (JS)

- Most popular language in the world!



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/2025s-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/2025s-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/2025s-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/2025s-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/2025s-cse467/demo/demo3.html

JS Example (3)

Overwrite whole document with 11

index.html

```html
<html>
  <script src=“write.js”>
  </script>
</html>
```

write.js

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

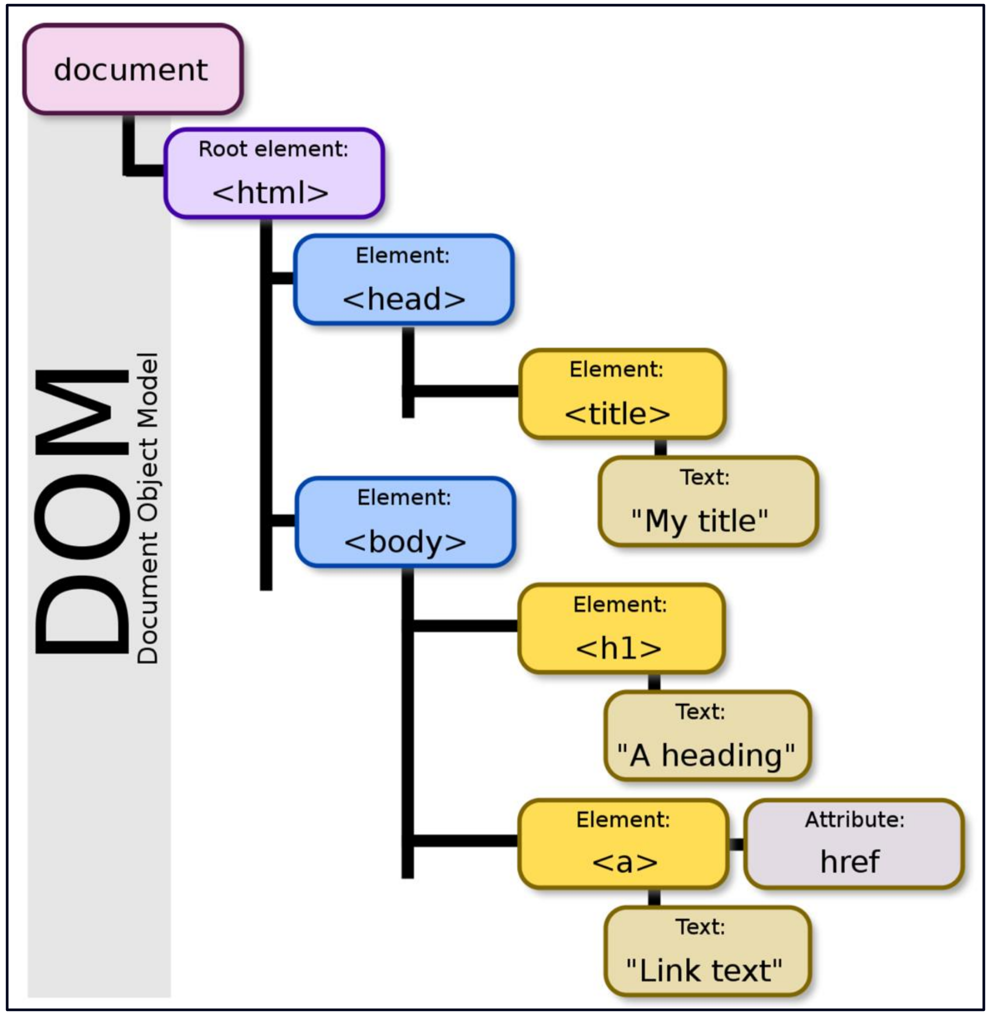
External script with `src` attribute

https://websec-lab.github.io/courses/2025s-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>
```



DOM and JS APIs

- Exposed to JavaScript through global objects
  - `document`: Access to the document (e.g., cookies, head/body)
  - `navigator`: Information about the browser (e.g., UA, plugins)
  - `screen`: Information about the screen (e.g., dimension, color depth)
  - `location`: Access to the URL (read and modify)
  - `history`: Navigation

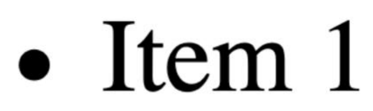
★ Changing HTML DOM using JS

- 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)

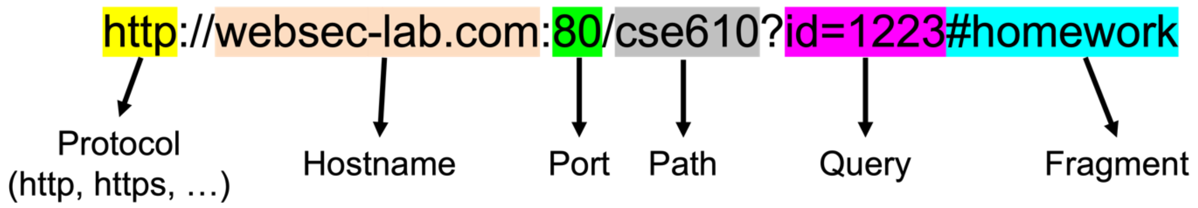
```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

- Item 1
- Item 2

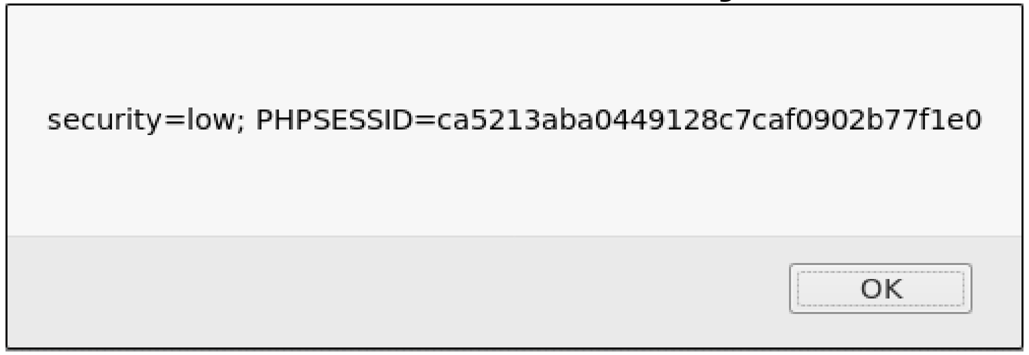
```
<script>
  var list = document.getElementById('t1’)
  var newitem = document.createElement('li’)
  var newtext = document.createTextNode(‘Item 2’)
  list.appendChild(newitem)
  newitem.appendChild(newtext)
</script>
```

Accessing HTML DOM using JS (Example)

- `location.protocol`: protocol
- `location.hostname`: only HTTP host
- `location.port`: only the port
- `location.pathname`: path


http://websec-lab.com:80/cse610?id=1223#homework
Protocol (http, https, …)
Hostname
Port
Path
Query
Fragment

- We can display all cookies for current document by `alert(document.cookie)`



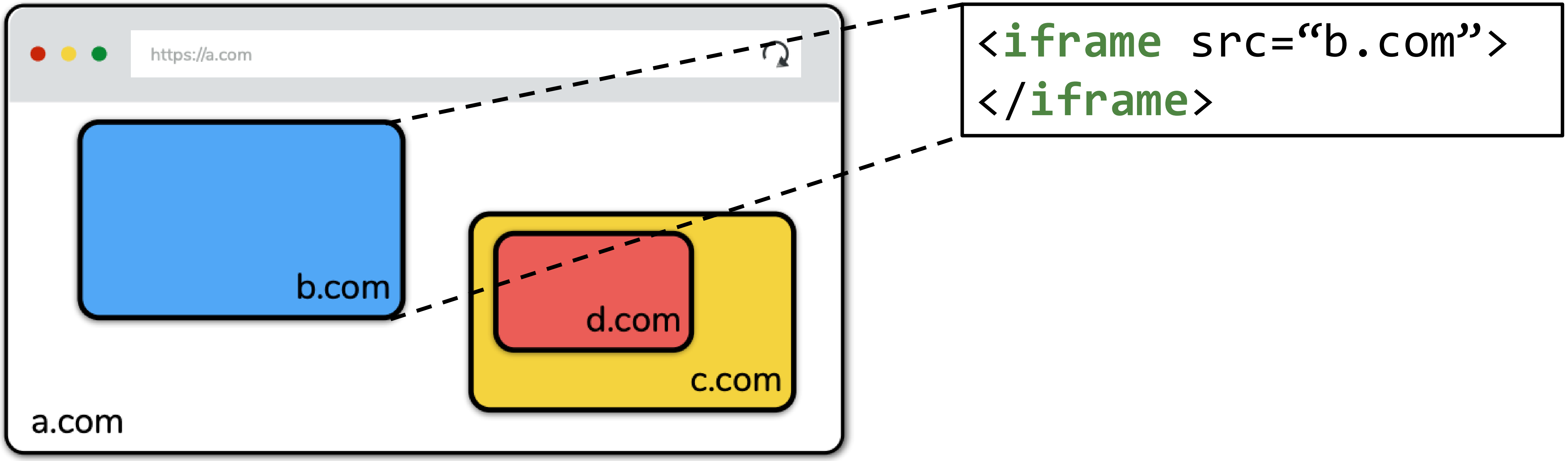
```
security=low; PHPSESSID=ca5213aba0449128c7caf0902b77f1e0
```

Basic Browser Execution Model

- Each browser window…
  - Fetches resources (e.g., images, CSS, Javascript)
  - Parses HTML and runs JavaScript
  - Loads content
  - 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

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**

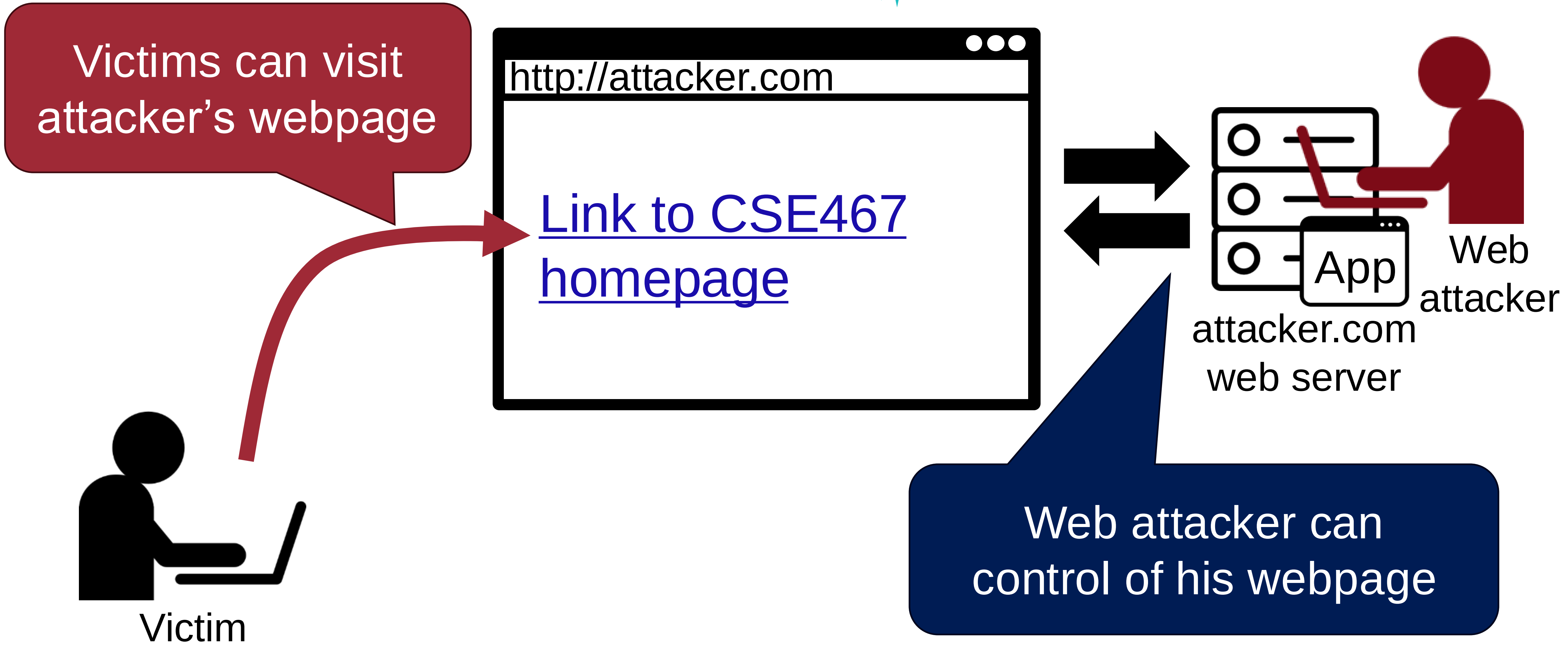
Web Threat Models

- **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



Question

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

Question?