Security and Privacy

SWE 432, Fall 2016

Design and Implementation of Software for the Web

GEORGE MASON UNIVERSITY

Today

- Security
  - What is it?
  - Most important types of attacks
- Privacy

For further reading:

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013
https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

Security

- Why is it important?
  - Users' data is on the web
    - Blog comments, FB, Email, Banking, …
  - Can others steal it?
    - or who already has access?
  - Can others impersonate the user?
    - e.g., post on FB on the user's behalf

Example

- https://gmu-swe432.github.io/lecture7demos/public/lecture7Demo2.html

Example

- Todo item:

/><script>alert("LASAGNA FOR PRESIDENT”);</script>

- Why does this work?

- What else might this be used to do?

Security Requirements for Web Apps

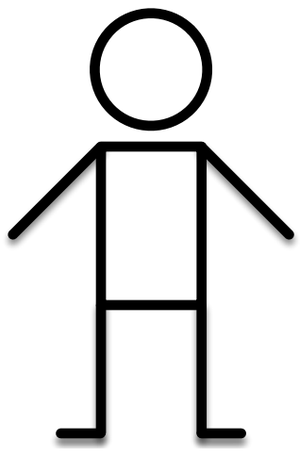
1. Authentication
   - Verify the **identify** of the parties involved
   - Who is it?
2. Authorization
   - Grant **access** to resources only to allowed users
   - Are you allowed?
3. Confidentiality
   - Ensure that **information** is given only to authenticated parties
   - Can you see it?
4. Integrity
   - Ensure that information is **not changed** or tampered with
   - Can you change it?

Threat Models

- What is being defended?
  - What resources are important to defend?
  - What malicious actors exist and what attacks might they employ?

- Who do we trust?
  - What entities or parts of system can be considered secure and trusted
  - Have to trust **something**!

Web Threat Models: Big Picture



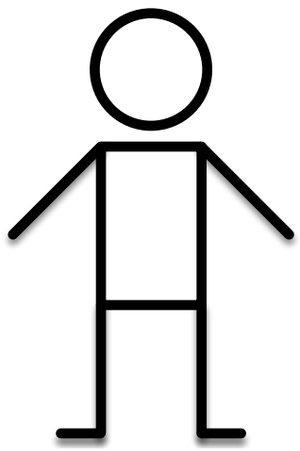
client page
(the “user”)

HTTP Request

HTTP Response

server

Web Threat Models: Big Picture



client page
(the “user”)

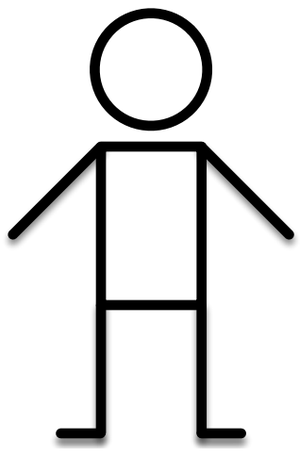
HTTP Request

HTTP Response

server

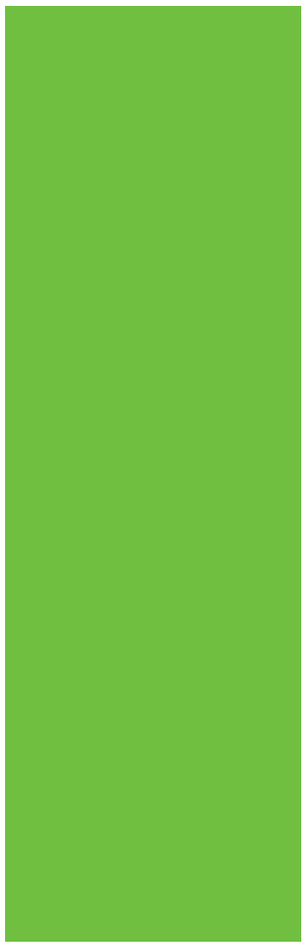
Do I trust that this request *really* came from the user?

Web Threat Models: Big Picture



HTTP Request

HTTP Response



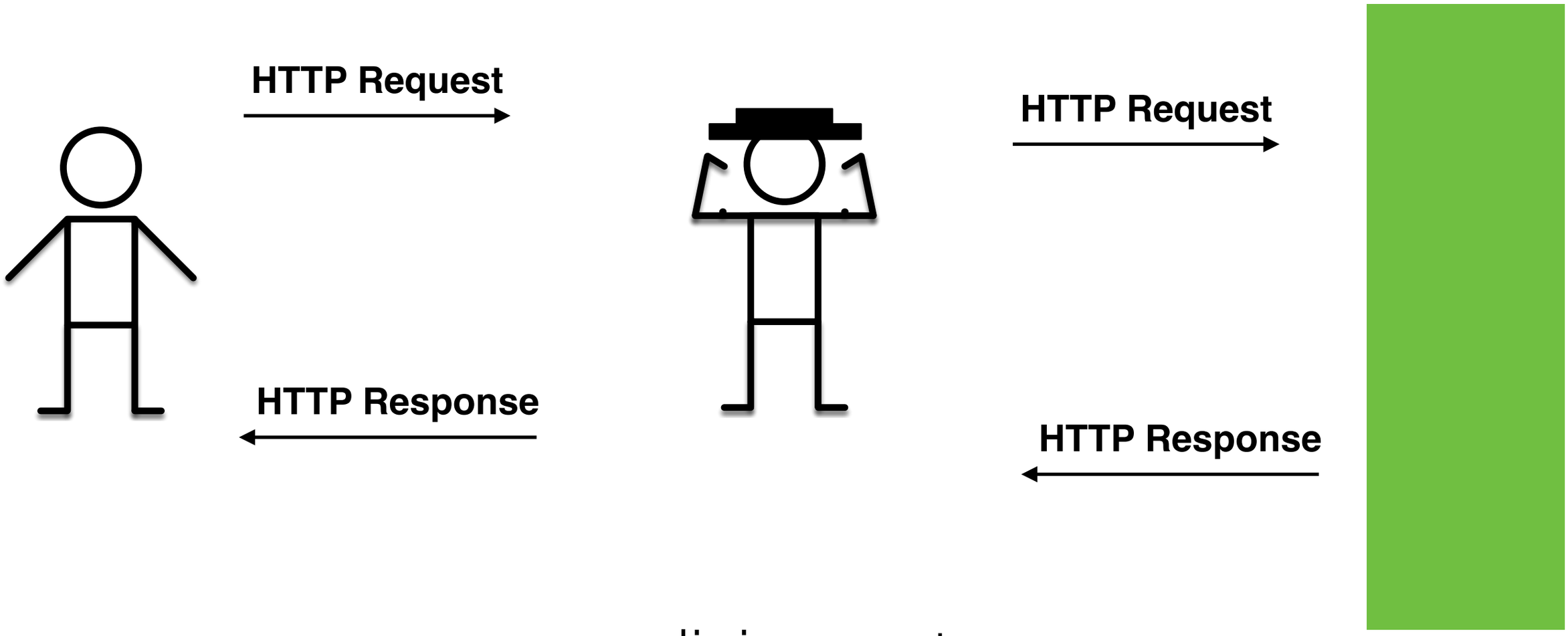
client page
(the “user”)

server

Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Web Threat Models: Big Picture



client page
(the "user")

malicious actor
"black hat"

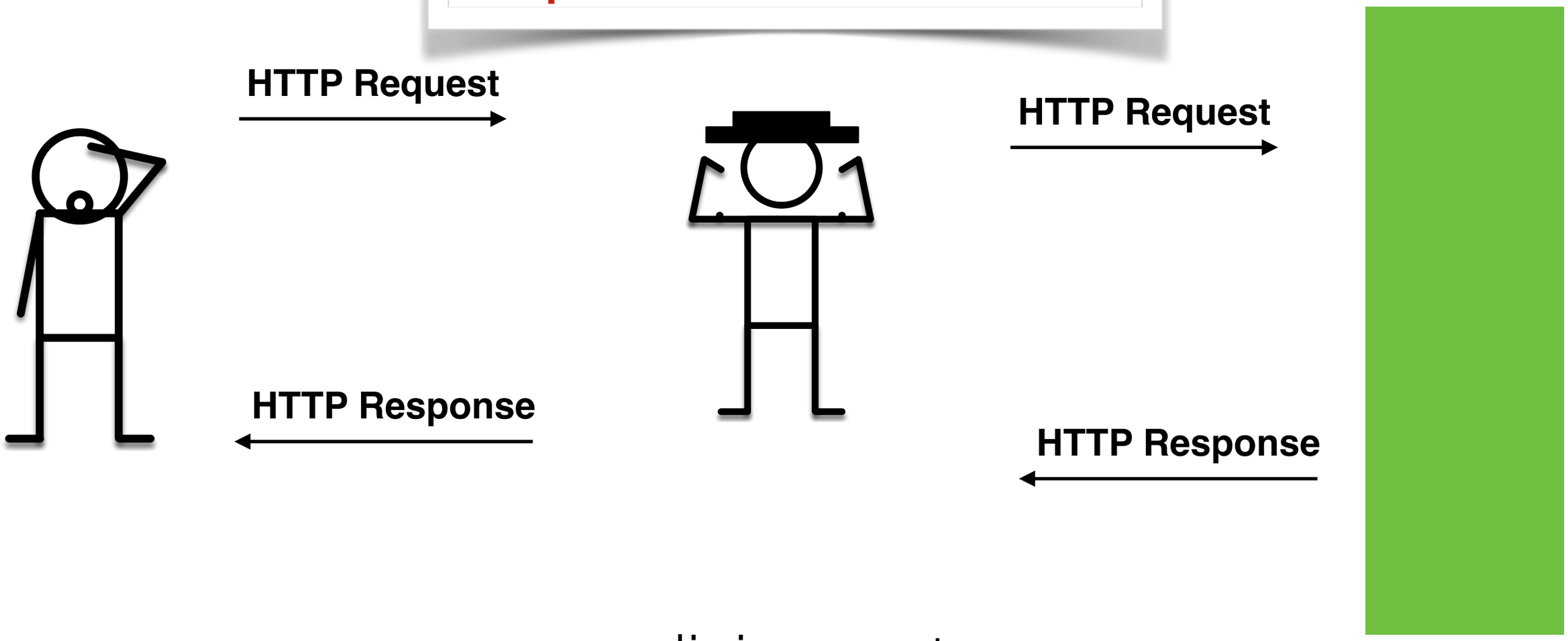
server

Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Web Threat Models: Big Picture

Might be "man in the middle" that intercepts requests and impersonates user or server.



client page (the "user")

malicious actor "black hat"

server

Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Security Requirements for Web Apps

1. Authentication
   - Verify the **identify** of the parties involved
   - Threat: Impersonation. A person pretends to be someone they are not.
2. Authorization
3. Confidentiality
   - Ensure that **information** is given only to authenticated parties
   - Threat: Eavesdropping. Information leaks to someone that should not have it.
4. Integrity
   - Ensure that information is **not changed** or tampered with
   - Threat: **Tampering**.

Sharing data between pages

- Browser loads many pages at the same time.
- Might want to share data between pages
  - Popup that wants to show details for data on main page
  - Cookies that let user login once for a page and still be logged in when visiting page in separate tab
- Attack: malicious page
  - User visits a malicious page in a second tab
  - Malicious page steals data from page or its cookies, modifies data, or impersonates user

Solution: Same-Origin Policy

- Browser needs to differentiate pages that are part of same application from unrelated pages
- What makes a page similar to another page?
  - Origin: the **protocol**, **host**, and **port**

http://www.example.com/dir/page.html

- Different origins:

https://www.example.com/dir/page.html

http://www.example.com:80/dir/page.html

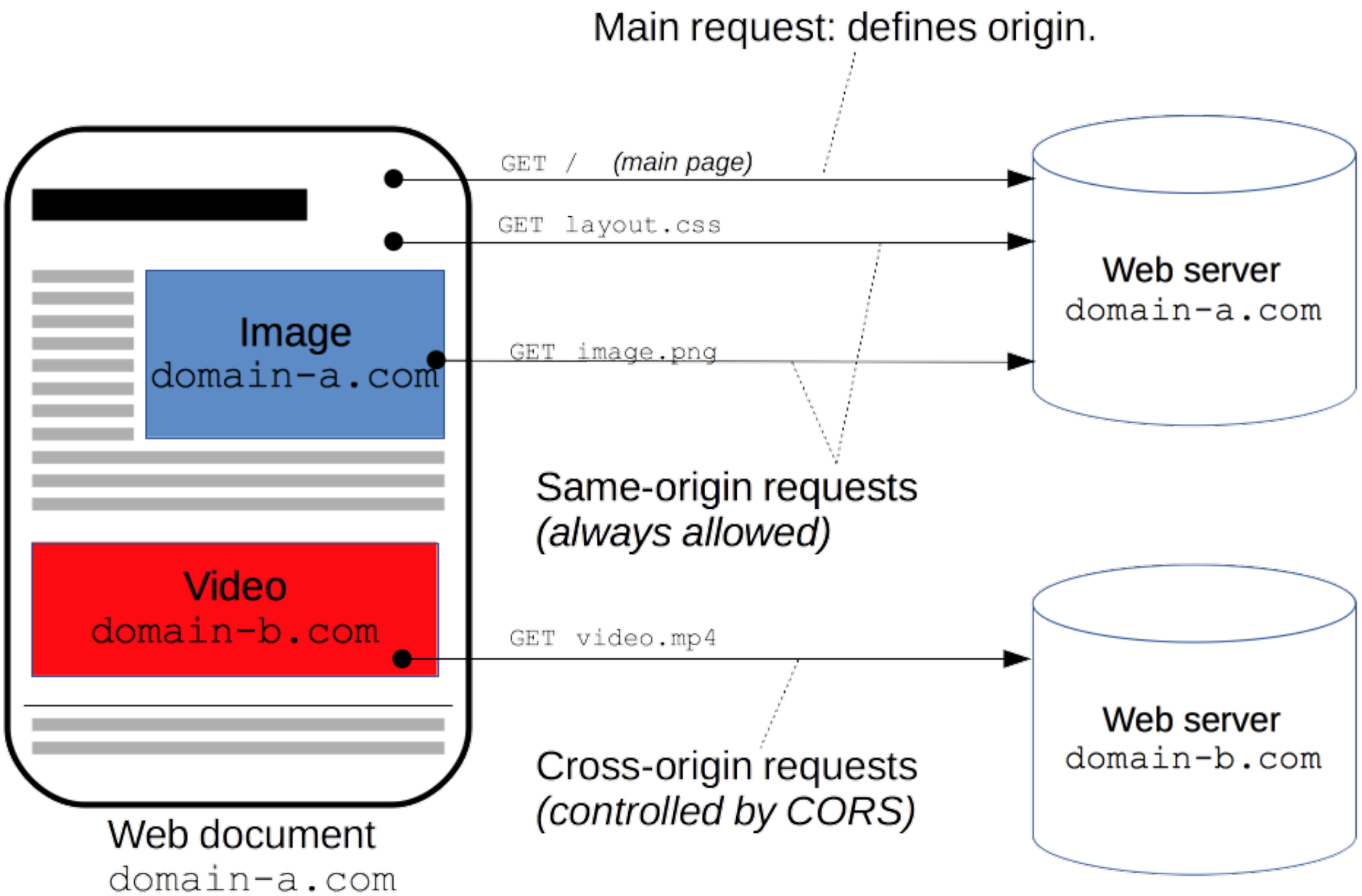
http://en.example.com:80/dir/page.html

https://en.wikipedia.org/wiki/Same-origin_policy

Same-Origin Policy

- "Origin" refers to the *page that is executing it*, NOT where the data comes from
  - Example:
    - In one HTML file, I directly include 3 JS scripts, each loaded from a different server
    - -> All have same "origin"
  - Example:
    - One of those scripts makes an AJAX call to yet another server
    - -> AJAX call not allowed
- Scripts contained in a page may access data in a second web page (e.g., its DOM) if they come from the same origin

Cross Origin Requests



https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

CORS: Cross Origin Resource Sharing

- Same-Origin might be safer, but not really usable:
  - How do we make AJAX calls to other servers?
- Solution: Cross Origin Resource Sharing (CORS)
- HTTP header:

```
Access-Control-Allow-Origin: <server or wildcard>
```

- In Express:

```
res.header("Access-Control-Allow-Origin", "*");
```

Top 3 Web Vulnerabilities

- OWASP collected data on vulnerabilities
  - Surveyed 7 firms specializing in web app security
  - Collected 500,000 vulnerabilities across hundreds of apps and thousands of firms
  - Prioritized by prevalence as well as exploitability, detectability, impact

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

#3 - XSS: Cross Site Scripting

- User input that contains a *client-side* script that does not belong
  - A todo item:

/><script>alert("LASAGNA FOR PRESIDENT”);</script>

- Works when user input is used to render DOM elements without being escaped properly
- User input saved to server may be served to other users
  - Enables malicious user to execute code on other’s users browser
  - e.g., click ‘Buy’ button to buy a stock, send password data to third party, …

#2 - Broken Authentication and Session Management

- Building authentication is hard
  - Logout, password management, timeouts, secrete questions, account updates, …
- Vulnerability may exist if
  - User authentication credentials aren't protected when stored using hashing or encryption.
  - Credentials can be guessed or overwritten through weak account management functions (e.g., account creation, change password, recover password, weak session IDs).
  - Session IDs are exposed in the URL (e.g., URL rewriting).
  - Session IDs don't timeout, or user sessions or authentication tokens, particularly single sign-on (SSO) tokens, aren't properly invalidated during logout.
  - Session IDs aren't rotated after successful login.
  - Passwords, session IDs, and other credentials are sent over unencrypted connections.

#1 - Injection

- User input that contains *server-side* code that does not belong
- Usually comes up in context of SQL (which we aren't using)
  - e.g.,

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

- Might come up in JS in context of eval
  - `eval(request.getParameter("code"));`
  - Obvious injection attack - don't do this!

Validating user input

- Escape Strings that originate from user
- Type of escaping depends on where data will be used
  - HTML - HTML entity encoding
  - URL - URL Escape
  - JSON - Javascript Escape
- Done automatically by some frameworks such as React
- More details: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Authentication

- How can we know the identify of the parties involved
- Want to customize experience based on identity
  - But need to determine identity first!
- Options
  - Ask user to create a new username and password
    - Lots of work to manage (password resets, storing passwords securely, …)
    - Hard to get right (#2 on the OWASP Top 10 Vulnerability List)
    - User does not really want another password…
  - Use an authentication provider to authenticate user
    - Google, FB, Twitter, Github, …

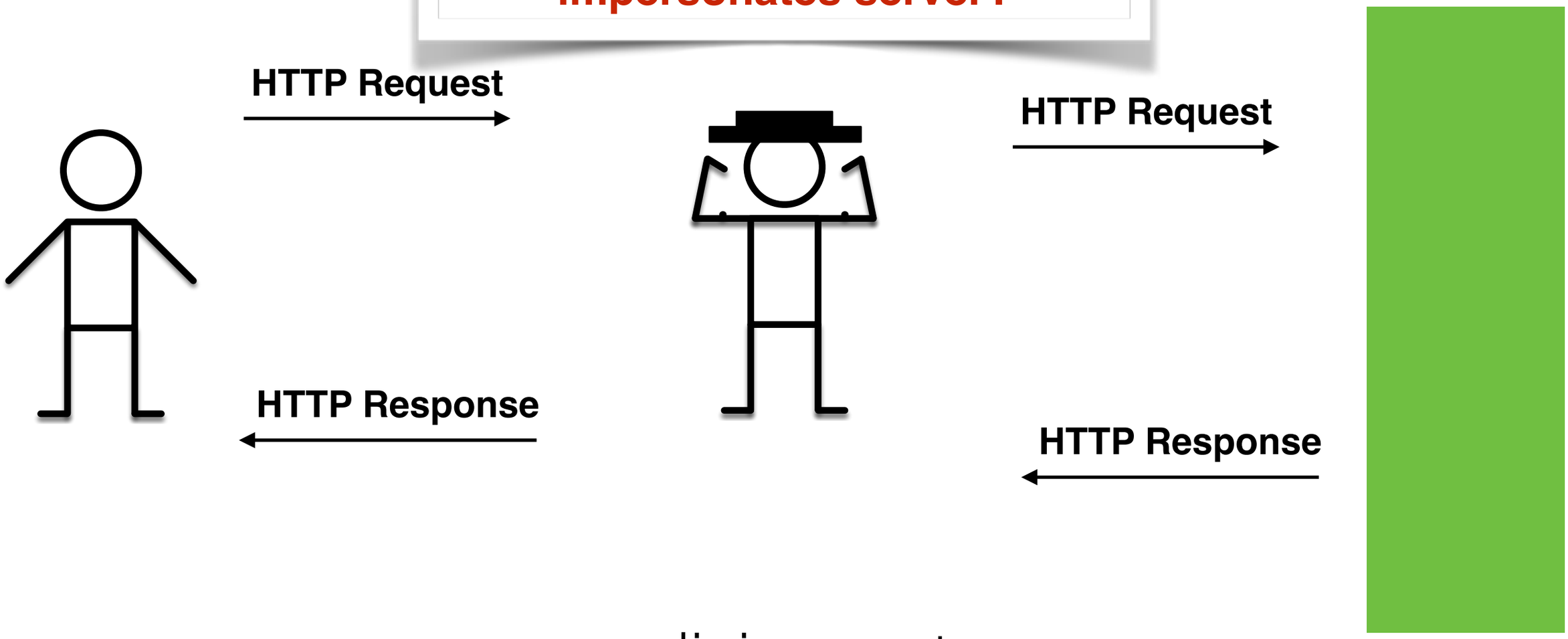
Authentication Provider

- Creates and tracks the identity of the user

- Instead of signing in directly to website, user signs in to authentication provider
  - Authentication provider issues token that uniquely proves identity of user
  - Talk more next lecture about how these tokens work

Web Threat Models: Big Picture

What if malicious actor impersonates server?



client page
(the “user”)

malicious actor
“black hat”

server

Man in the middle

- Requests to server intercepted by man in the middle
  - Requests forwarded
  - But… response containing code edited, inserting malicious code
- Or could
  - Intercept and steal sensitive user data

HTTPS: HTTP over SSL

- Establishes secure connection from client to server
  - Uses SSL to encrypt traffic
- Ensures that others can't impersonate server by establishing certificate authorities that vouch for server.
- Server trusts an HTTPS connection iff
  - The user trusts that the browser software correctly implements HTTPS with correctly pre-installed certificate authorities.
  - The user trusts the certificate authority to vouch only for legitimate websites.
  - The website provides a valid certificate, which means it was signed by a trusted authority.
  - The certificate correctly identifies the website (e.g., certificate received for "https://example.com" is for "example.com" and not other entity).

Using HTTPS

- If using HTTPS, important that all scripts are loaded through HTTPS
  - If mixed script from untrusted source served through HTTP, attacker could still modify this script, defeating benefits of HTTPS
- Example attack:
  - Banking website loads jQuery through HTTP from a CDN rather than HTTPS
  - Attacker intercepts request for jQuery script, replaces with malicious script that steals user data or executes malicious action

Privacy

- What does it mean to you?

Some definitions of privacy

- Secrecy
- Contextual integrity
- Limited access to the self
- Control over information

Adapted from Conceptions of Privacy by Lorrie Faith Cranor

Limited access to self

"the right to be let alone"

- Samuel D. Warren and Louis D. Brandeis,
The Right to Privacy,
4 Harv. L. Rev. 193 (1890)

"Our concern over our accessibility to others: the extent to which we are **known to others**, the extent to which others have **physical access** to us, and the extent to which we are the **subject of others attention**.

- Ruth Gavison, "Privacy and the Limits of the Law," Yale Law Journal 89 (1980)

Control over information

"Privacy is the claim of individuals, groups or institutions to determine for themselves when, how, and to what extent information about them is communicated to others."

"…each individual is continually engaged in a personal adjustment process in which he balances the desire for privacy with the desire for disclosure and communication…."

Alan Westin, Privacy and Freedom, 1967

Data

- What sorts of data might be collected
  - The products you browsed on Amazon
  - The emails you wrote or were sent to you
  - The location of where you were last Monday
  - Everything you've bought with a credit card

Who might use this data?

- Service owner
  - Recommendations: based on your past history, you are like to want to watch
- Data might be shared with third parties (e.g., advertisers)
  - Based on an email you just received from your family, you are likely to want to buy a plane ticket to Seattle.
- Even aggregate data can have value
- How do you decide what data to share or not?

Mechanisms for enforcing privacy

- Do Not Track: http://donottrack.us/
  - Proposed W3C standard implemented by current browsers
  - Aims to give users control by letting them opt out of tracking

- Hard to implement without regulatory enforcement
  - Nothing today (in US) to ensure that websites follow user preference
  - Other countries (e.g., EU) starting to develop stronger legal protections for privacy

https://en.wikipedia.org/wiki/Do_Not_Track

Takeaways

- Think about all potential threat models
  - Which do you care about
  - Which do you not care about
- What user data are you retaining
  - Who are you sharing it with, and what might they do with it