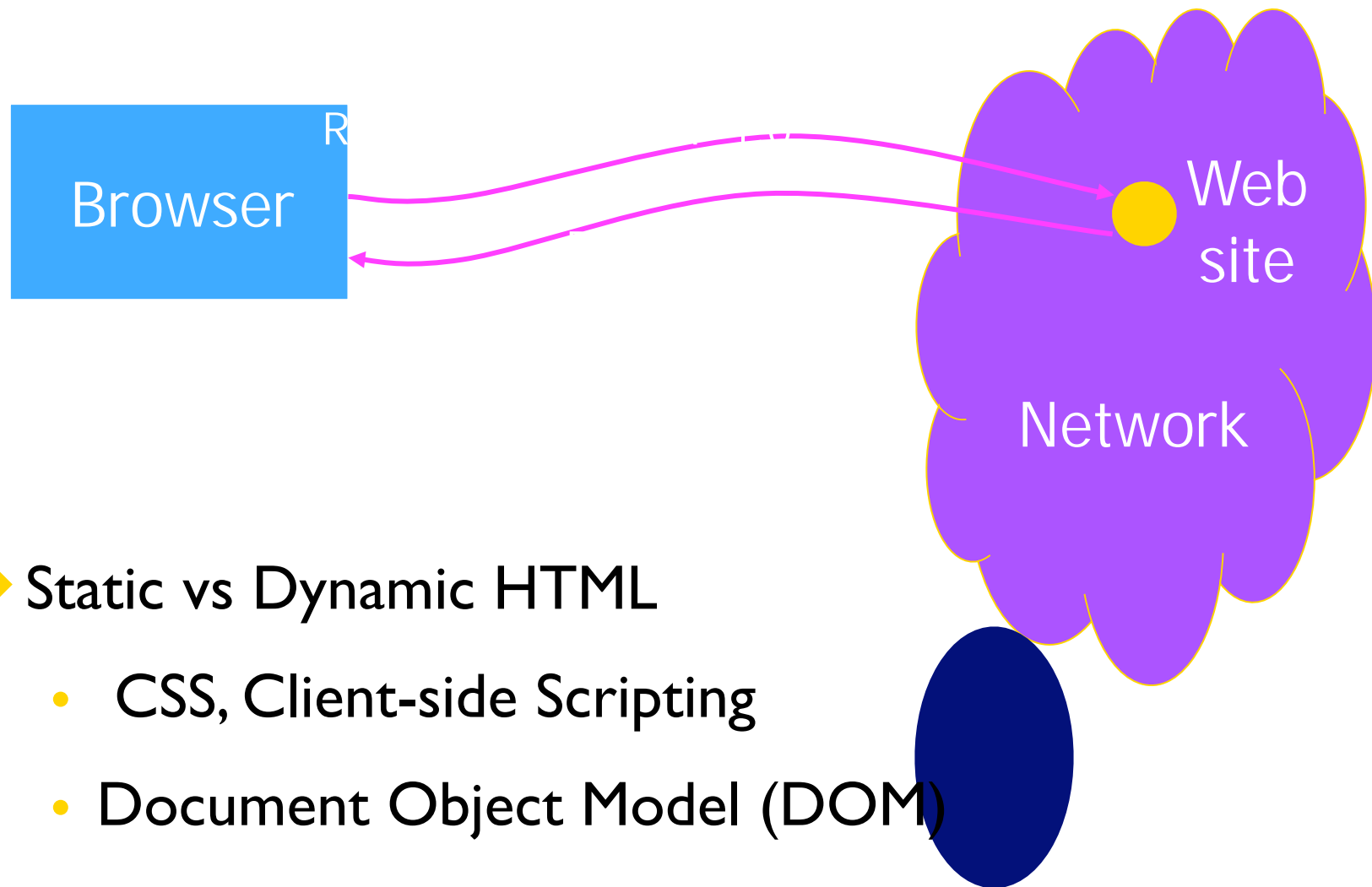


# **Cross Domain Security in Web Applications**

# Browser and Network



## ◆ Static vs Dynamic HTML

- CSS, Client-side Scripting
- Document Object Model (DOM)

## ◆ Scripts can interact with page elements, and completely change page elements

# DOM and scripts

```
<ul>
```

```
<li name="hey"> ...</item>
```

```
....
```

```
</ul>
```

```
<a
```

```
onmouseover=document.getElementById('hey').style.fontSize='10'> Make it Tiny!</a>
```

```
<a
```

```
onmouseover=document.getElementById('hey').style.fontSize='24'> Make it HUGE!!!</a>
```

# Web Browsers and Same origin

- Scripts can only access properties associated with documents of the same origin
- Same origin
  - Cookies
  - DOM objects and Attributes
- \_How is origin defined?
  - Host name
  - Protocol (HTTP, FTP)
  - Port
- Note that document URL is \*not\* part of origin

# Same Origin Policy

<http://www.examplesite.org/here>

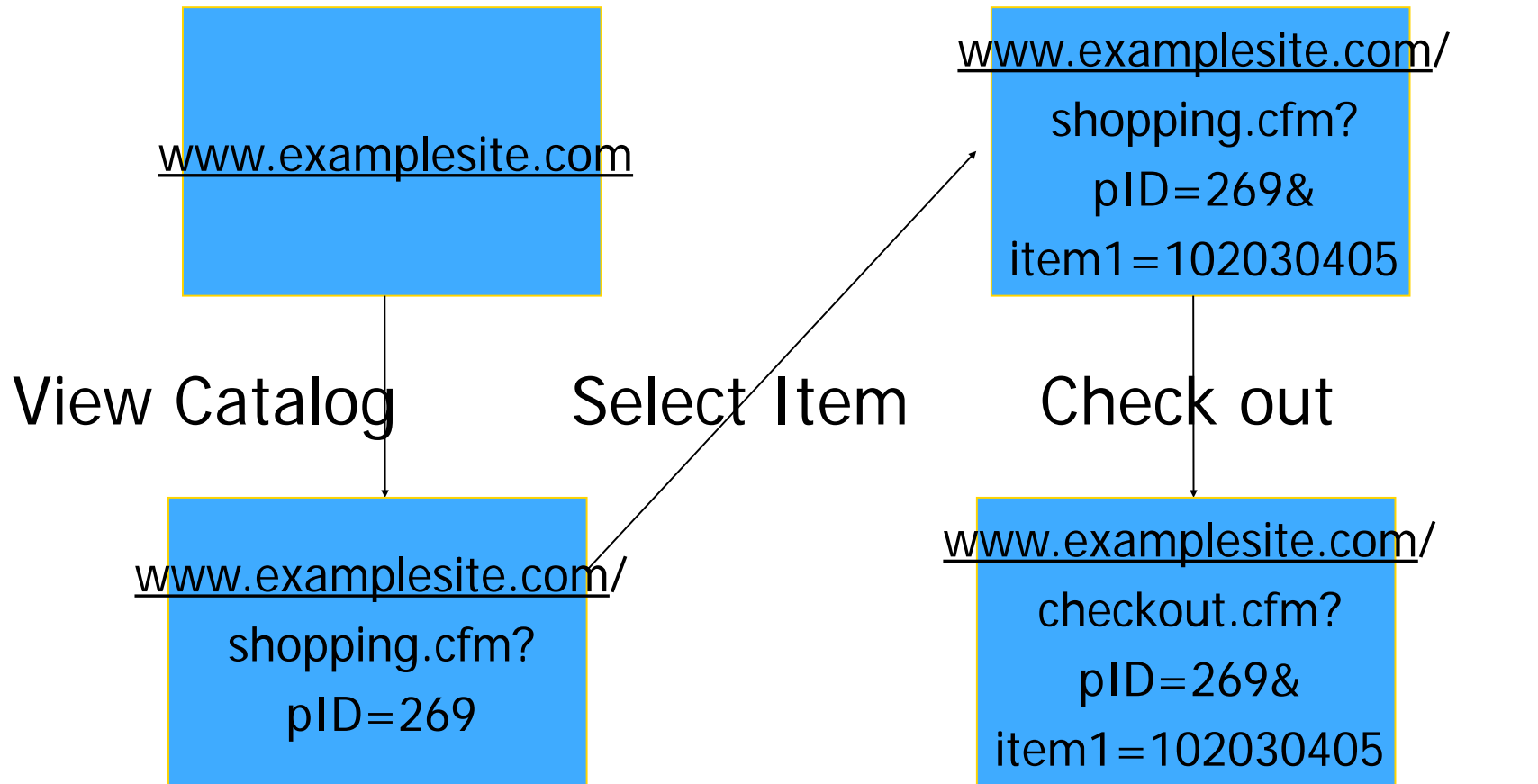
<https://www.examplesite.org/there>

<http://www.examplesite.com:8080/thar>

<http://www.hackerhome.org/yonder>

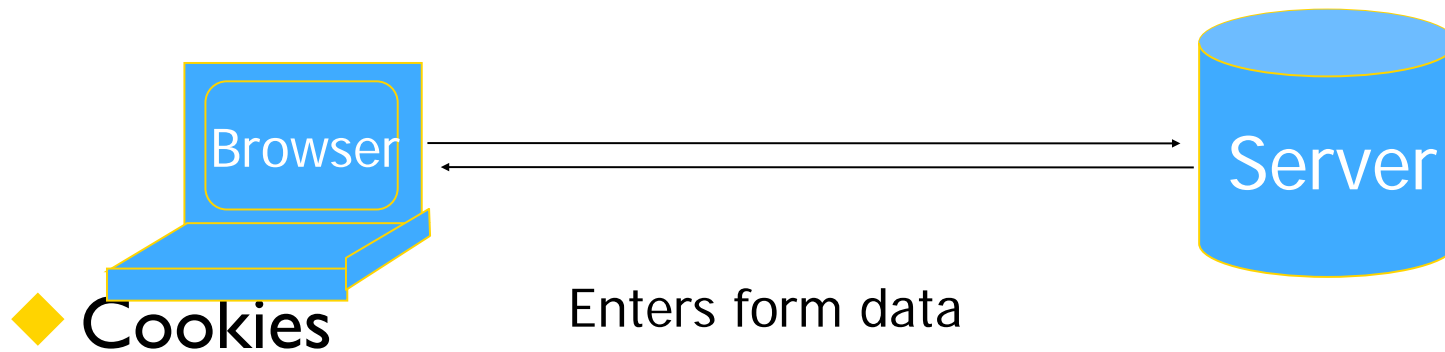
Which of these are part of the same origin?

# Basic Browser Session

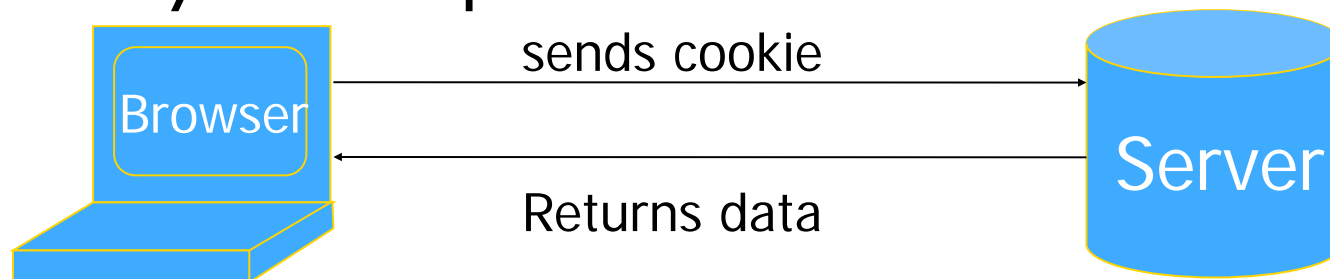


Store session information in URL; Easily read on network

# Store info across sessions?



- A cookie is a piece of data (in fact, a file) created by an Internet site to store information on your computer



Http is stateless protocol; cookies add state

# Interactions between different origins

- Same origin prevents scripts from [www.hackerhome.org](http://www.hackerhome.org) from accessing a page from [www.examplesite.com](http://www.examplesite.com)
- Say reading [examplesite.com](http://examplesite.com)'s cookie
- However, several other forms of interaction exist between the domains
- Interactions happen because linking is a fundamental pattern on the web

# Interactions

- Suppose user loads a page from
  - www.hackerhome.org which contains a link to  
`<a href=http://www.examplesite.com>Click here</a>`
- Say user subsequently clicks on this
  - Will be redirected to the page
  - Link controlled by attacker
  - Web doesn't control who can link

# Site Interactions

- This is possible
  - ``  
(pixel)
    - User probably does not notice
- Worse
  - `<iframe style="display:none" src=http://www.examplesite.com>Click here</iframe>`

# Same Origin Policy

- Previous example, hackerhome.org cannot peek into examplesite.com's frame
- Same origin policy prevents this
- Even if frame is embedded in a browser window

# XSRF attacks

## Login Name/Password Retrieval and Account Reminder

Please provide your E-mail Address or Login Name. An e-mail containing your Login Name and Password, and your subscription information will be sent to the e-mail address on record for your account.

**Enter Your E-mail Address**

AND/OR

**Enter Your Login Name**

**Send my Login Name and Password**

# Example Application code - changing passwords

```
<form method="POST" action="/  
  update_profile">
```

...

```
New Password: <input type="password"  
  name="password">
```

```
</form>
```

Application subsequently determines identity  
using a cookie (stored earlier)

# Attack page

- Suppose attacker lures the client to visit his /her web page

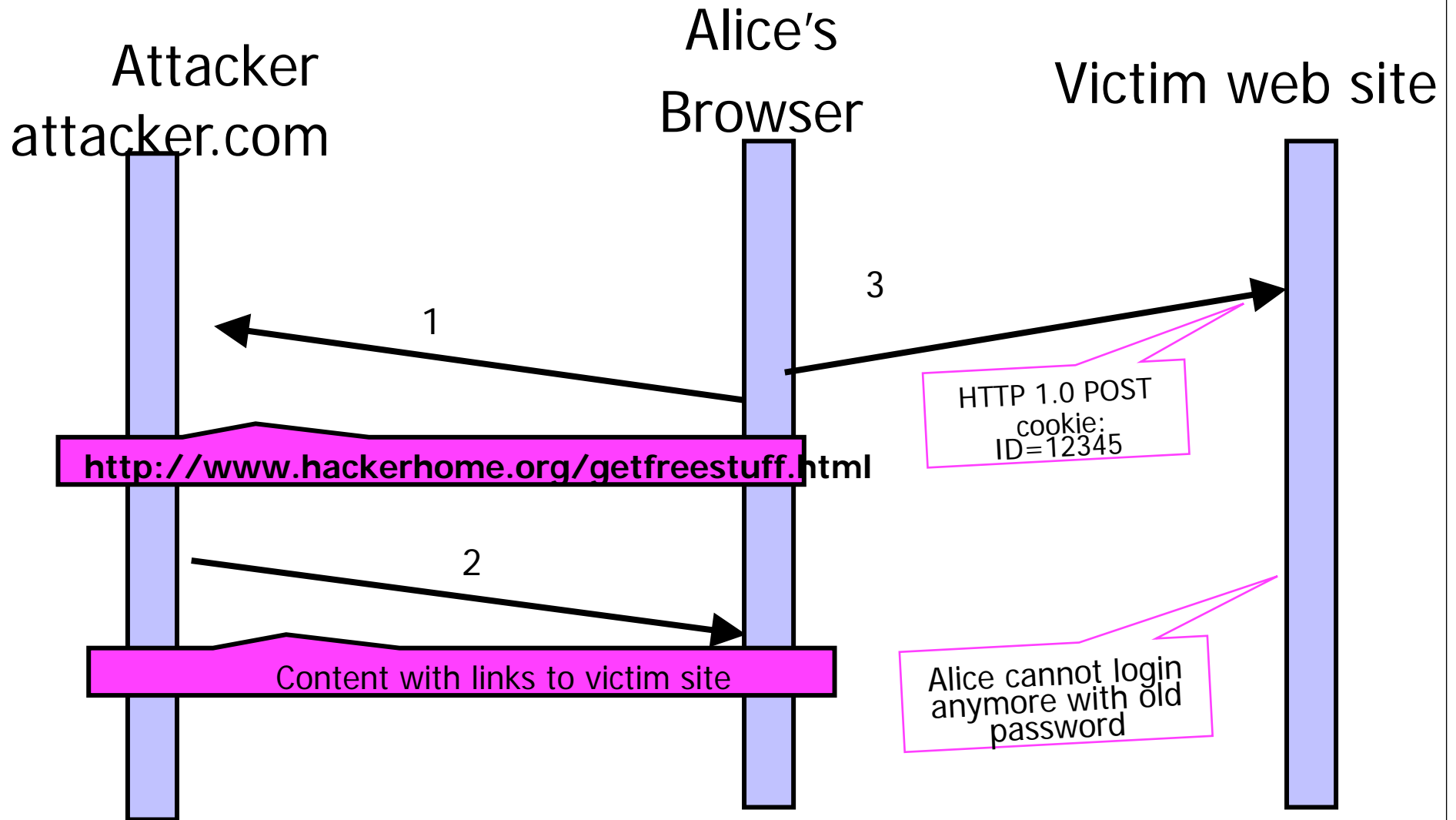
```
<form method="POST" name="evilform"  
  target="hiddenframe" action=https://  
  www.examplesite.com/update_password>
```

```
<input type="hidden" name="password"  
  value="evilhax0r">
```

```
</form>
```

```
<iframe name="hiddenframe" style="display:none"></  
  iframe>
```

# Forged Requests



# More details

- Alice cannot login anymore to the website
  - Note that attacker doesn't get any cookie
- Applications with features that allow users to update profile info
  - Bulletin board messages
  - E-commerce applications
  - Any application that stores data on behalf of a user

# An attack to steal cookies

- ◆ Cross-site scripting
- ◆ Instance of an input validation attack
- ◆ (similar to SQL injection where improper input validation leads to attack)
- ◆ Results in disclosure of cookie information to attacker