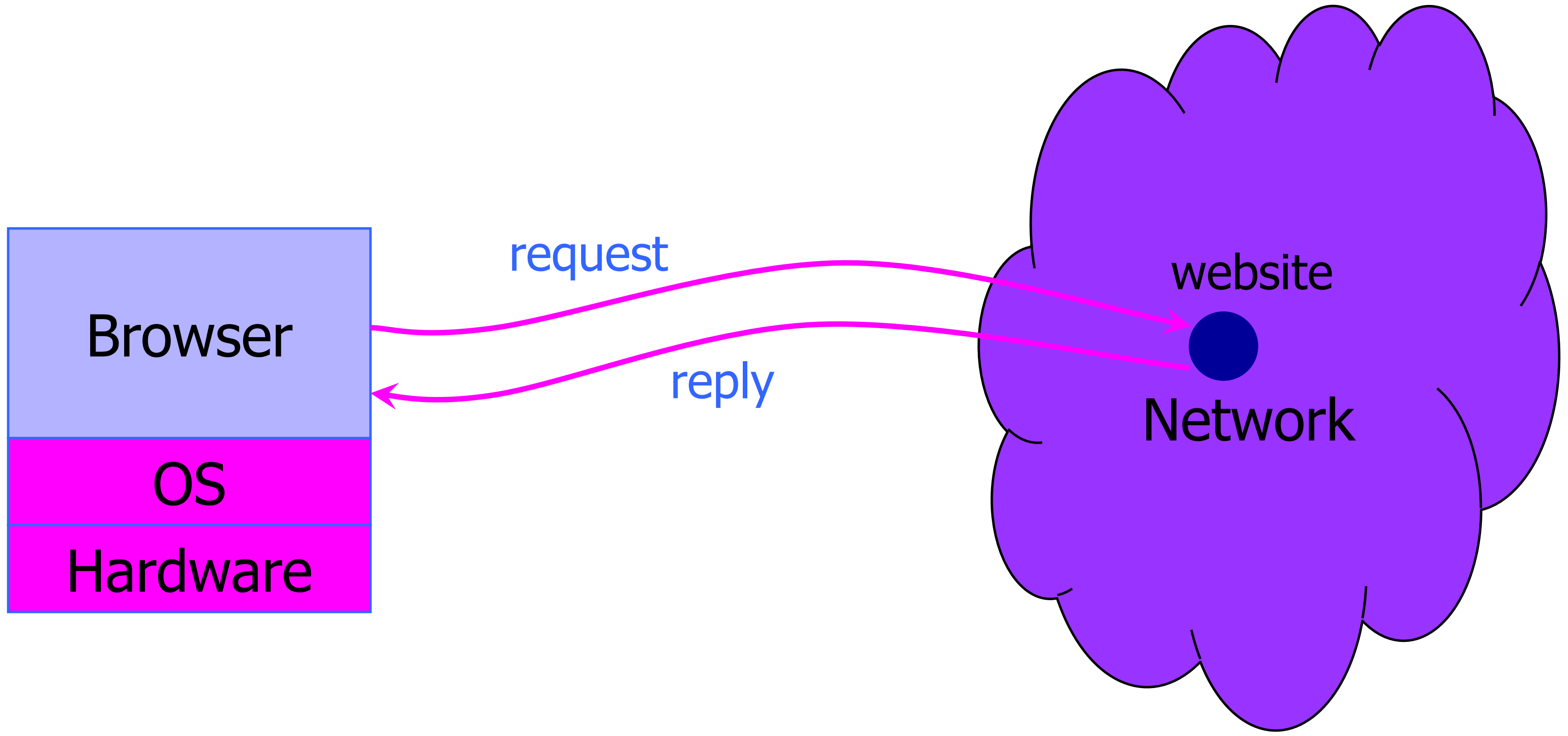
CS 5450

HTTP

Vitaly Shmatikov

Browser and Network



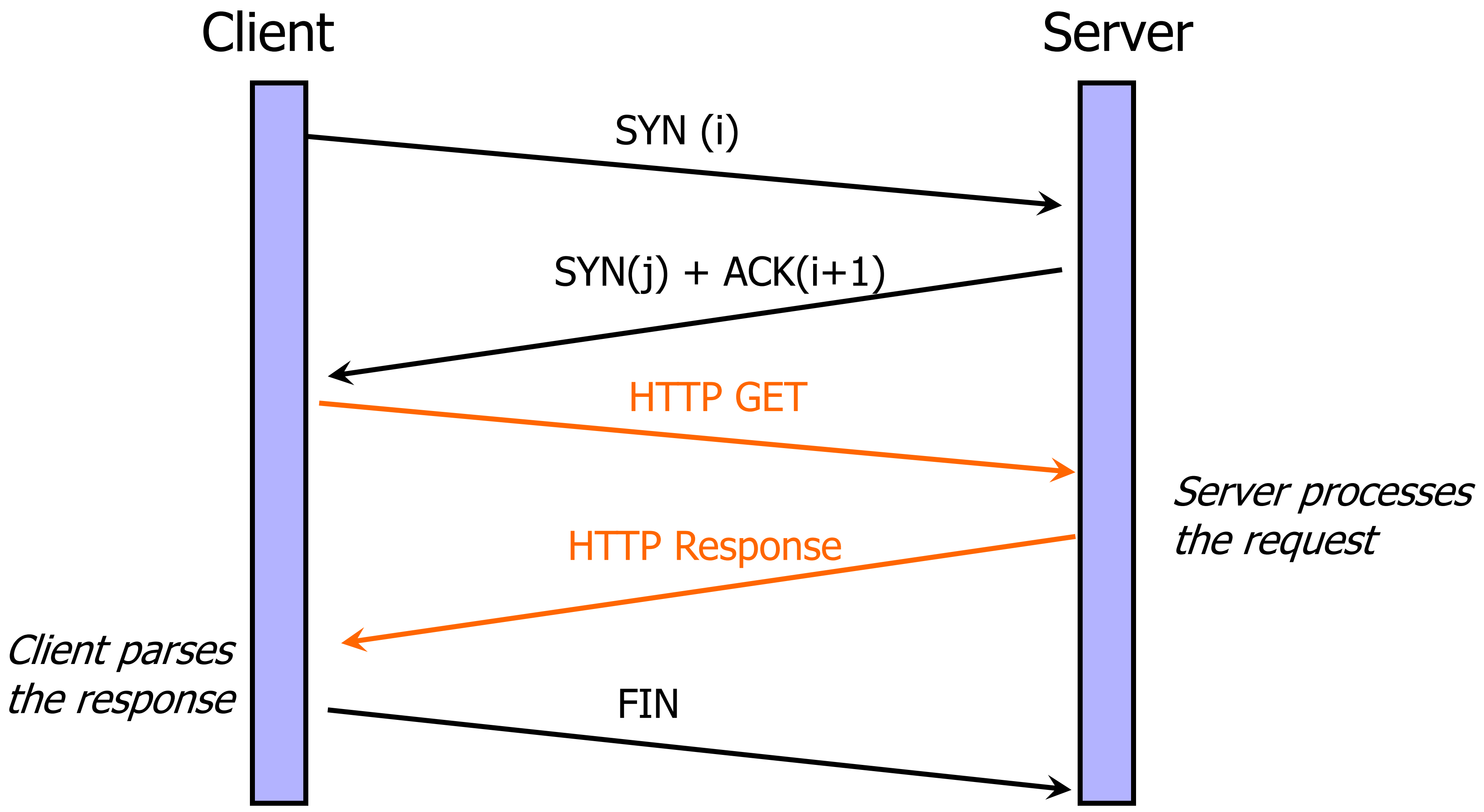
HTML

- A web page includes
  - Base HTML file
  - Referenced objects (e.g., images)
- HTML: Hypertext Markup Language
  - Representation of hypertext documents in ASCII
  - Web browsers interpret HTML when rendering a page
    - Format text, reference images, embed hyperlinks (HREF)

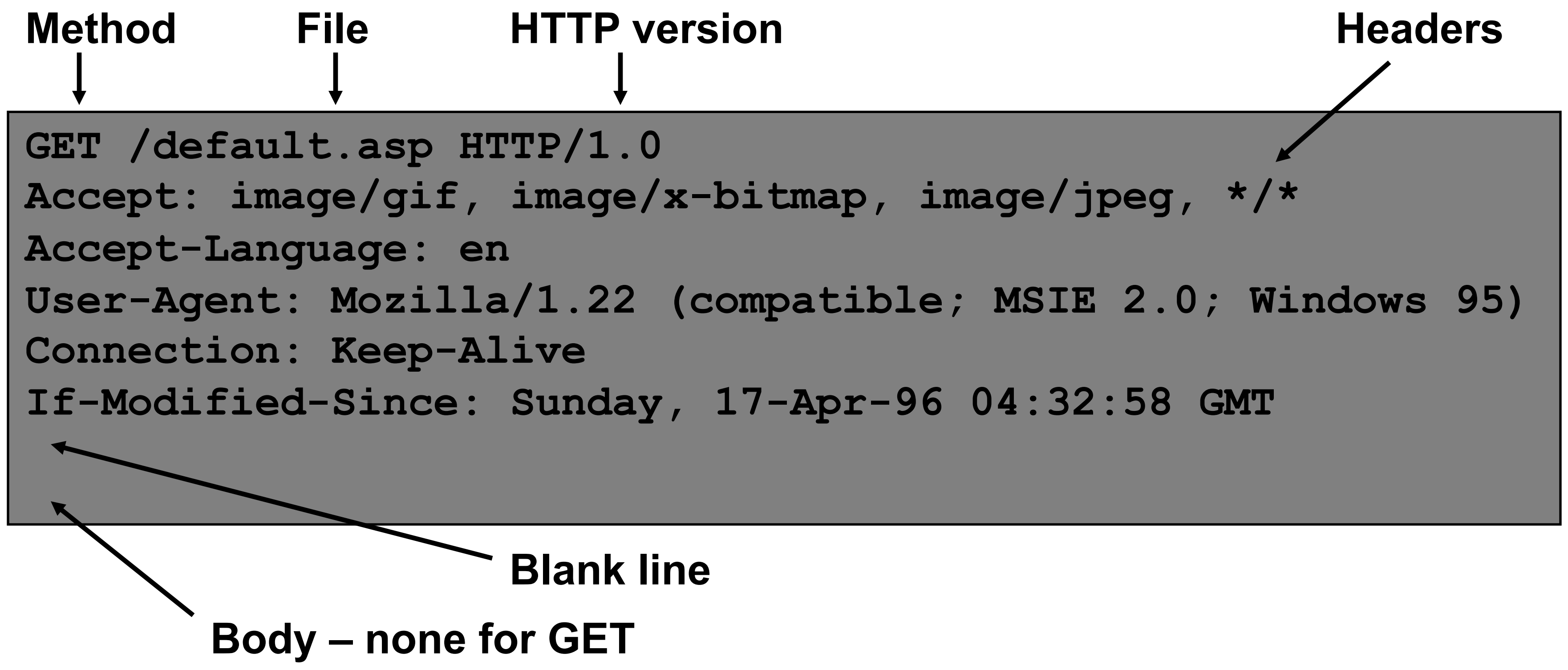
HTTP: HyperText Transfer Protocol

- Used to request and return data
  - Methods: GET, POST, HEAD, …
- Stateless request/response protocol
  - Each request is independent of previous requests
  - Statelessness has a significant impact on design and implementation of applications
- Evolution
  - HTTP 1.0: simple
  - HTTP 1.1: more complex

Steps in an HTTP Request



HTTP Request



```
GET /default.asp HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Connection: Keep-Alive
If-Modified-Since: Sunday, 17-Apr-96 04:32:58 GMT

```

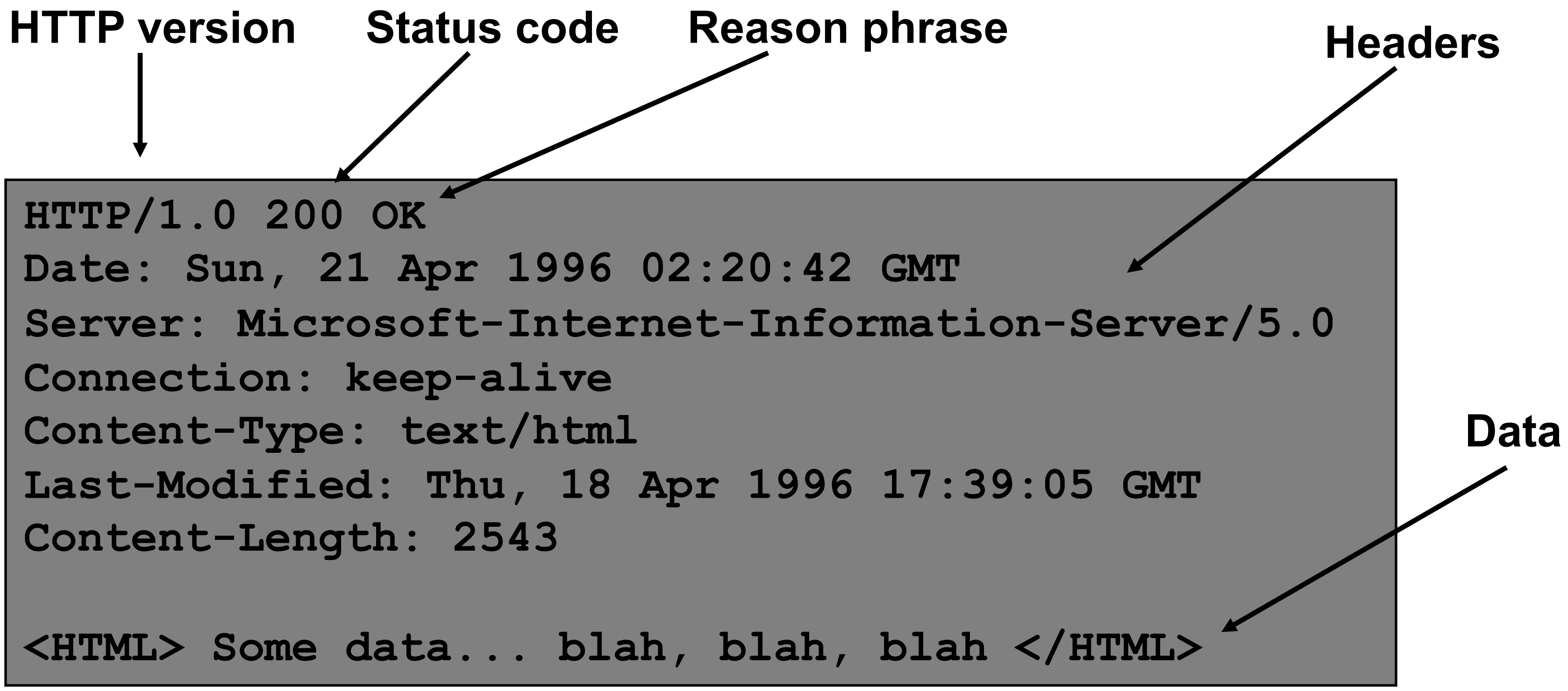
HTTP Methods

- GET
  - Return current value of a resource, run program, etc.
- HEAD
  - Return the metadata associated with a resource
- POST
  - Update resource, provide input to a program

Generating a Response

- Return a file ("static content")
  - URL matches a file (e.g., /www/index.html)
  - Server returns the file's contents as the response
  - Server generates appropriate response header
- Generate a response dynamically
  - URL triggers a program on the server
  - Server executes the program and sends output to client in the HTTP response
  - Return metadata with no body

HTTP Response



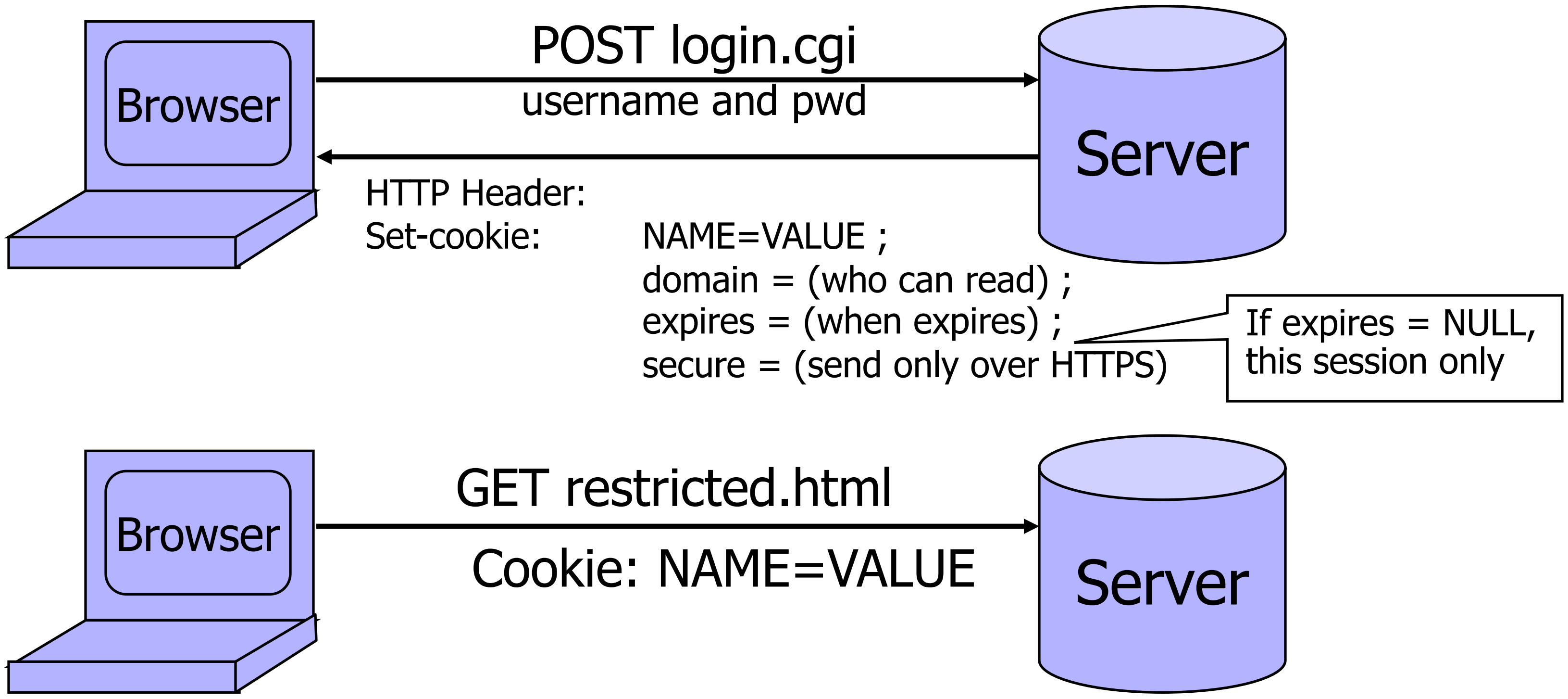
```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543

<HTML> Some data... blah, blah, blah </HTML>
```

HTTP Is Stateless

- Each request-response exchange is treated independently, server not required to retain state
- Good: improves scalability on the server side
  - Don’t have to retain info across client requests
  - Can handle higher rate of client requests
  - Order of client requests doesn’t matter
- Bad: some apps need persistent state
  - Uniquely identify user or store temporary info (e.g., shopping cart, user preferences/profiles, usage tracking, etc.)

Website Storing Info In Browser

A cookie is a file created by a website to store information in the browser



HTTP is a stateless protocol; cookies add state

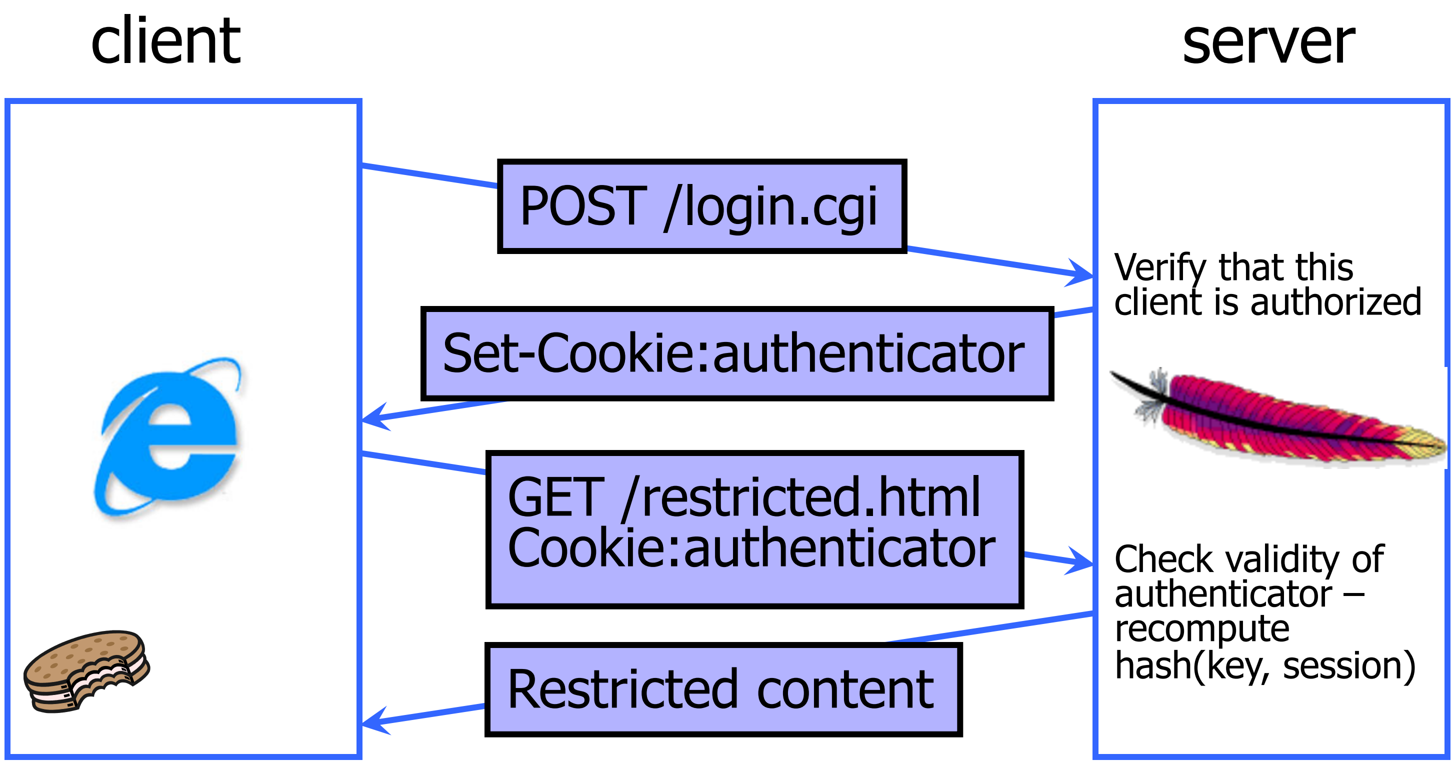
What Are Cookies Used For?

- Authentication
  - The cookie proves to the website that the client previously authenticated correctly
- Personalization
  - Helps the website recognize the user from a previous visit
- Tracking
  - Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Web Authentication with Cookies

- Authentication system that works over HTTP and does not require servers to store session data
  - … except for logout status
- After client successfully authenticates, server computes an authenticator token and gives it to the browser as a cookie
  - Client should not be able forge authenticator on his own
    - Example: HMAC(server's secret key, session information)
- With each request, browser presents the cookie; server recomputes and verifies the authenticator
  - Server does not need to remember the authenticator

Typical Session with Cookies

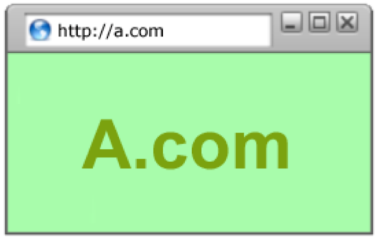


Authenticators must be unforgeable and tamper-proof

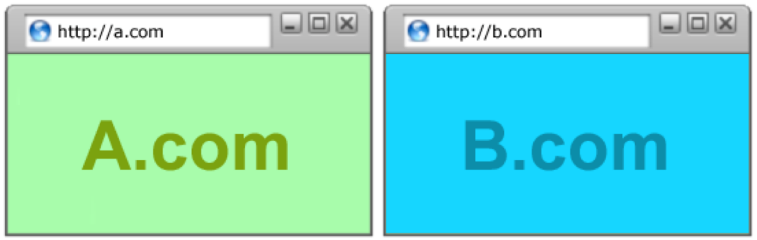
(malicious client shouldn't be able to compute his own or modify an existing authenticator)

Goals of Browser Security

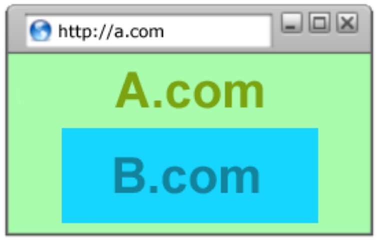
- Safe to visit an evil website



- Safe to visit two pages at the same time



- Safe delegation



Browser: Basic Execution Model

- Each browser window or frame:
  - Loads content
  - Renders
    - Processes HTML and scripts to display the page
    - May involve images, subframes, etc.
  - Responds to events
- Events
  - User actions: OnClick, OnMouseover
  - Rendering: OnLoad, OnUnload

JavaScript

- "The world's most misunderstood programming language"
- Language executed by the browser
  - Scripts are embedded in Web pages
  - Can run before HTML is loaded, before page is viewed, while it is being viewed, or when leaving the page
- Used to implement "active" web pages
  - AJAX, huge number of Web-based applications

JavaScript History

- Developed by Brendan Eich at Netscape
  - Scripting language for Navigator 2
- Later standardized for browser compatibility
  - ECMAScript Edition 3 (aka JavaScript 1.5)
- Related to Java in name only
  - Name was part of a marketing deal
  - “Java is to JavaScript as car is to carpet”
- Various implementations available

JavaScript in Web Pages

- Embedded in HTML page as <script> element
  - JavaScript written directly inside <script> element
    - <script> alert("Hello World!") </script>
  - Linked file as src attribute of the <script> element

    <script type="text/JavaScript" src="functions.js"></script>
- Event handler attribute

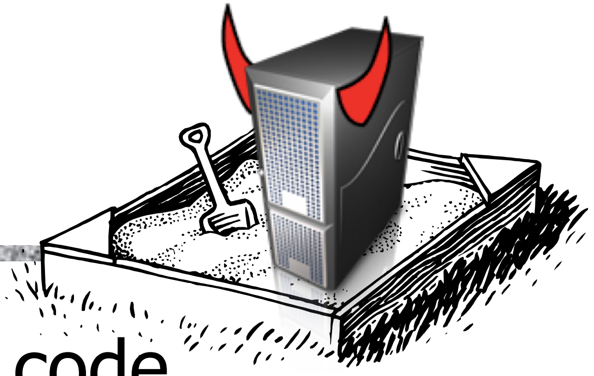
  <a href="http://www.yahoo.com" onmouseover="alert('hi');">
- Pseudo-URL referenced by a link

  <a href="JavaScript: alert('You clicked');">Click me</a>
- Many other ways

Document Object Model (DOM)

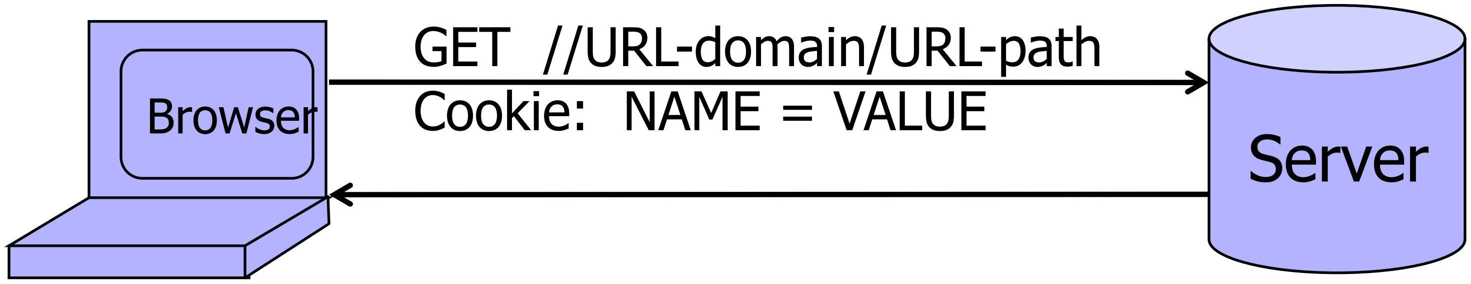
- HTML page is structured data
- DOM is object-oriented representation of the hierarchical HTML structure
  - Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], …
  - Methods: document.write(document.referrer)
    - These change the content of the page!
- Also Browser Object Model (BOM)
  - Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Browser Sandbox



- Goal: safely execute JavaScript code provided by a remote website
  - No direct file access, limited access to OS, network, browser data, content that came from other websites
- Same origin policy (SOP)
  - Can only read properties of documents and windows from the same <u>protocol</u>, <u>domain</u>, and <u>port</u>

SOP for Reading Cookies



Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain
- cookie-path is prefix of URL-path
- protocol=HTTPS if cookie is “secure”

Examples of Cookie Reading SOP

cookie 1
name = **userid**
value = u1
domain = **login.site.com**
path = **/**
secure

cookie 2
name = **userid**
value = u2
domain = **.site.com**
path = **/**
non-secure

both set by **login.site.com**

http://checkout.site.com/	cookie: userid=u2
http://login.site.com/	cookie: userid=u2
https://login.site.com/	cookie: userid=u1; userid=u2

SOP for JavaScript in Browser

- Same domain scoping rules as for sending cookies to the server
- document.cookie returns a string with all cookies available for the document
  - Often used in JavaScript to customize page
- Javascript can set and delete cookies via DOM
  - document.cookie = "name=value; expires=…; "
  - document.cookie = "name=; expires= Thu, 01-Jan-70"