CIS 551 / TCOM 401

Computer and Network Security

Spring 2007

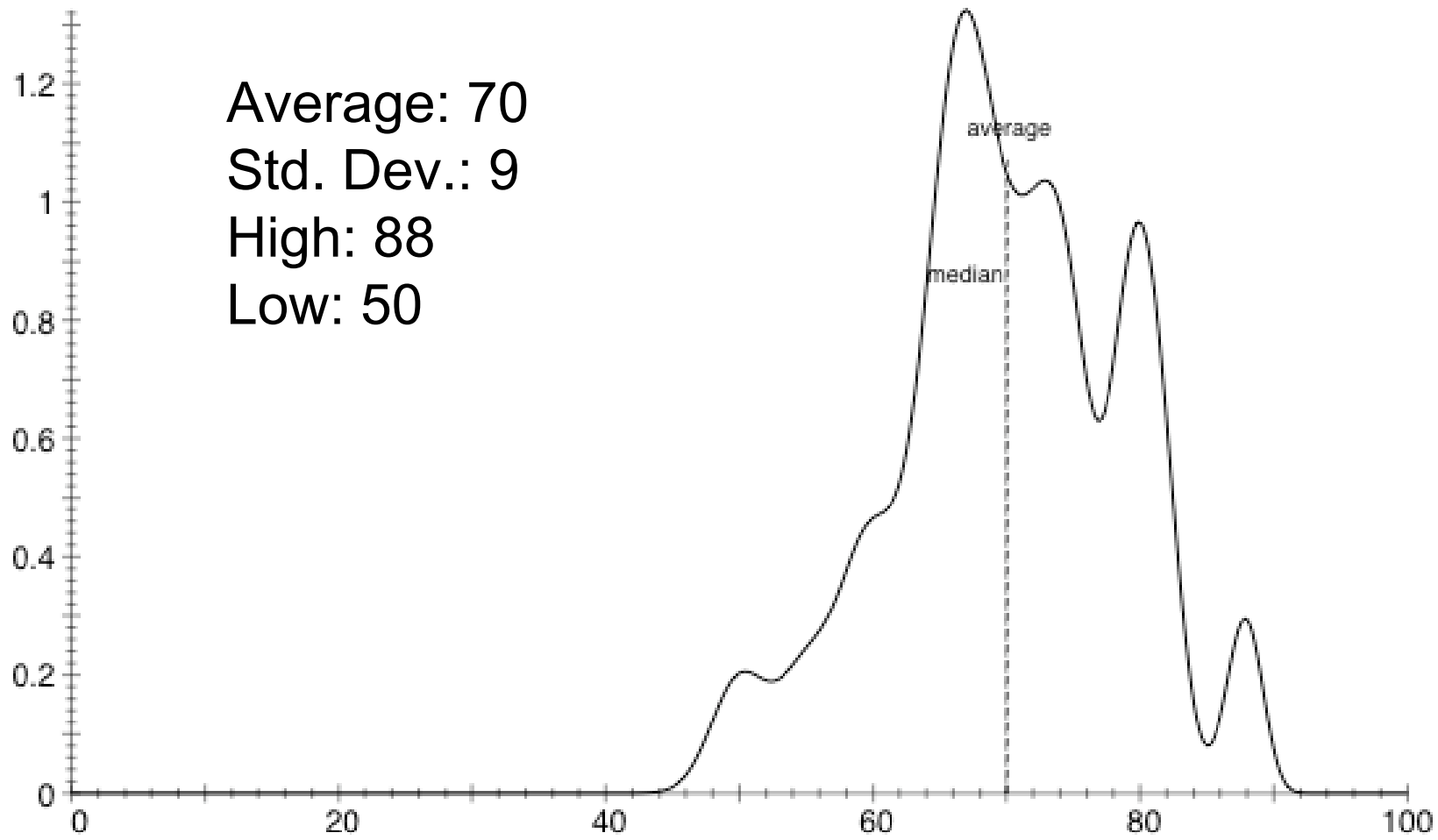
Lecture 20

Announcements

- Reminder: Project 3 is available on the web pages
  - Due: April 3rd
- Midterm II has been graded
- Today:
  - Web Security
  - [Some slides adapted from John Mitchell's course at Stanford]

Midterm II

Average: 70
Std. Dev.: 9
High: 88
Low: 50



Web Security

- What security concerns are there on the web?

- Class answers:
  - Leaking personal information (privacy!)
    - Anonymity
  - Integrity -- getting data/software from the web
    - Web scripts, web pages can modify local data
  - Authenticating remote hosts -- Phishing / spoofing
  - SQL injection / XSS / format string vulerabilities
  - Cookies / encoding state in URLs

OWASP.org Top 10

- Open Web Application Security Project

1. Unvalidated Input
2. Broken Access Control
3. Broken Authentication
4. Cross Site Scripting
5. Buffer Overflows
6. Injection Flaws
7. Improper Error Handling
8. Insecure Storage
9. Application Denial of Service
10. Insecure Configuration Management

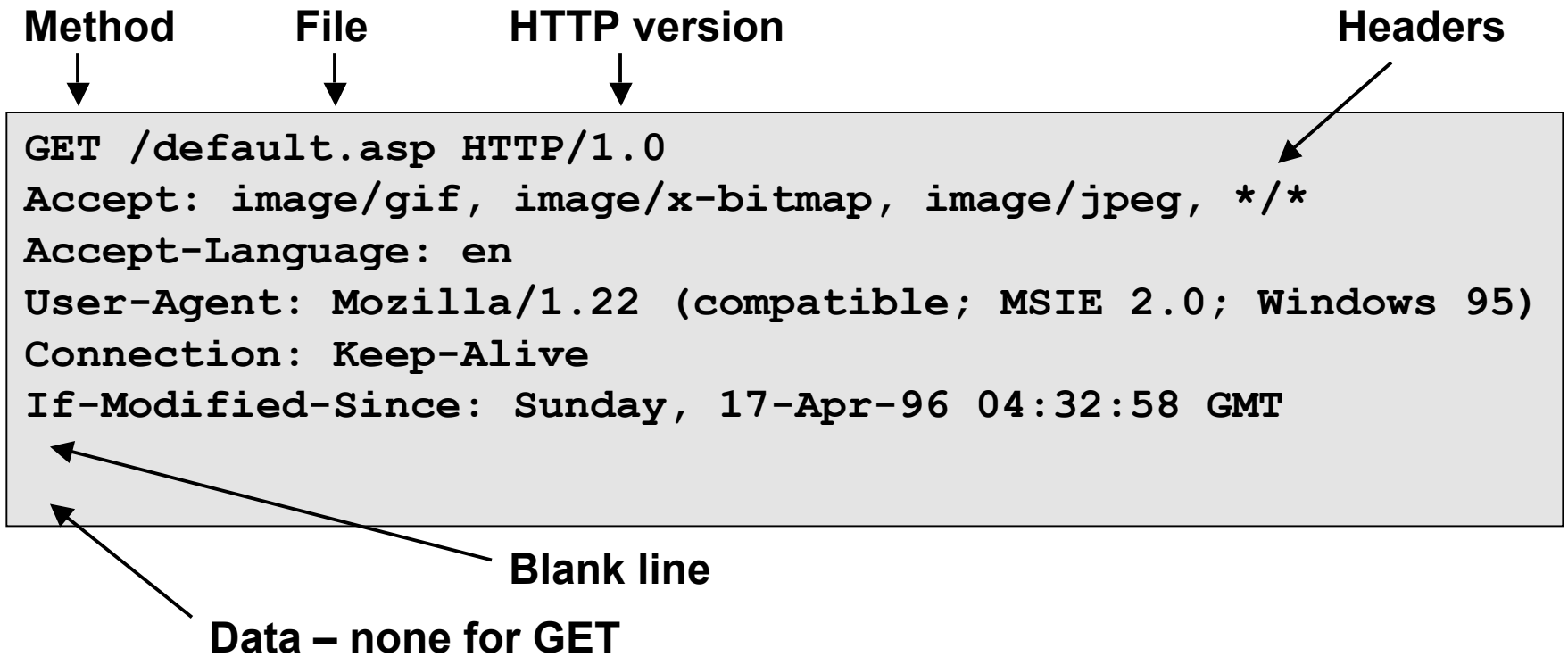
Browser security topics

- Review HTTP, scripting
- Controlling outgoing information
  - Cookies
    - Cookie mechanism, JunkBuster
  - Routing privacy
    - Anonymizer, Crowds
  - Privacy policy – P3P
- Risks from incoming executable code
  - JavaScript
  - ActiveX
  - Plug-ins
  - Java

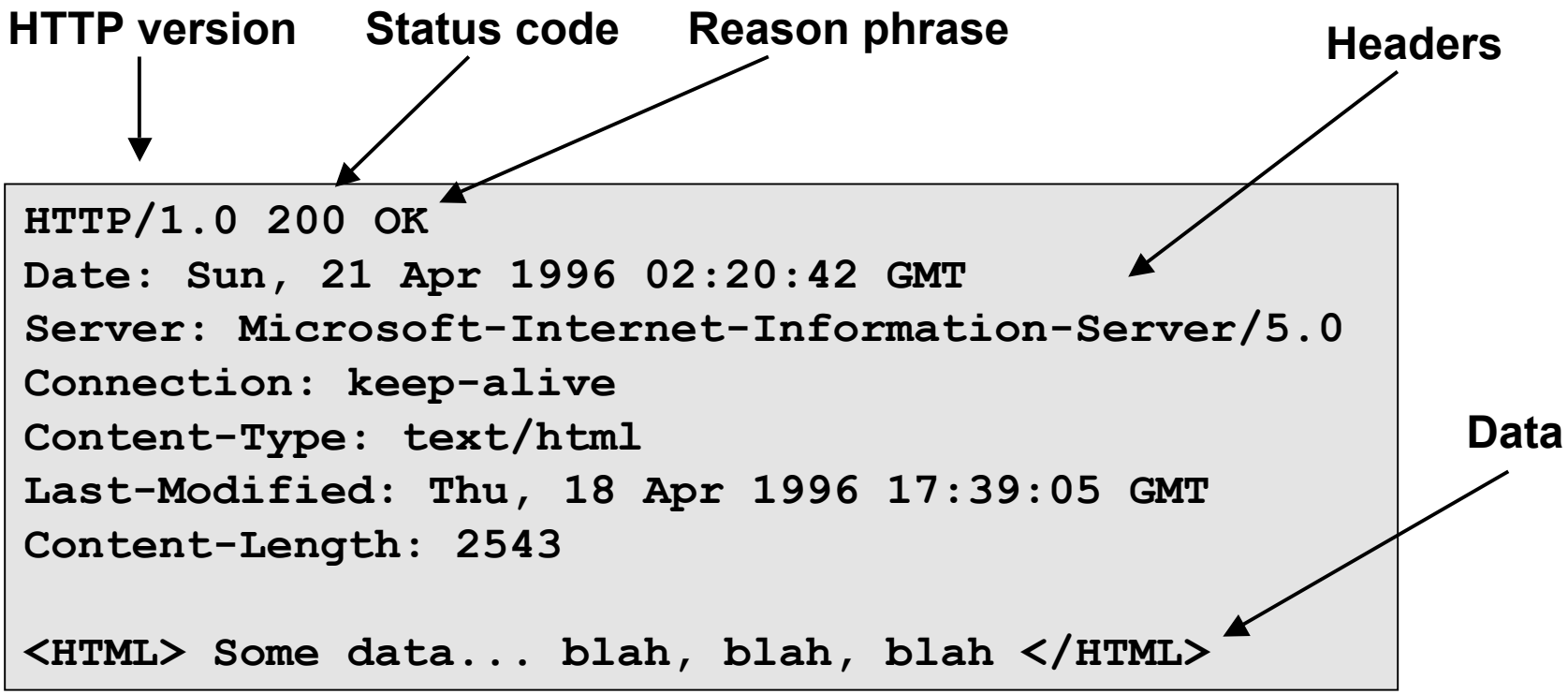
HyperText Transfer Protocol

- 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

HTTP Request



HTTP Response



HTTP Server Status Codes

Code	Description
200	OK
201	Created
301	Moved Permanently
302	Moved Temporarily
400	Bad Request – not understood
401	Unauthorized
403	Forbidden – not authorized
404	Not Found
500	Internal Server Error

- Return code 401
  - Used to indicate HTTP authorization
  - HTTP authorization has serious problems!!!

HTML and Scripting

```
<html>
      …
    <P>
<script>
      var num1, num2, sum
      num1 = prompt("Enter first number")
      num2 = prompt("Enter second number")
      sum = parseInt(num1) + parseInt(num2)
      alert("Sum = " + sum)
</script>
      …
</html>
```

Browser receives content, displays HTML and executes scripts

Events

```
<script type="text/javascript">
    function whichButton(event) {
      if (event.button==1) {
                alert("You clicked the left mouse button!") }
      else {
                alert("You clicked the right mouse button!")
      }}
</script>
…
<body onmousedown="whichButton(event)">
…
</body>
```

Mouse event causes page-defined function to be called

Other events: onLoad, onMouseMove, onKeyPress, onUnLoad

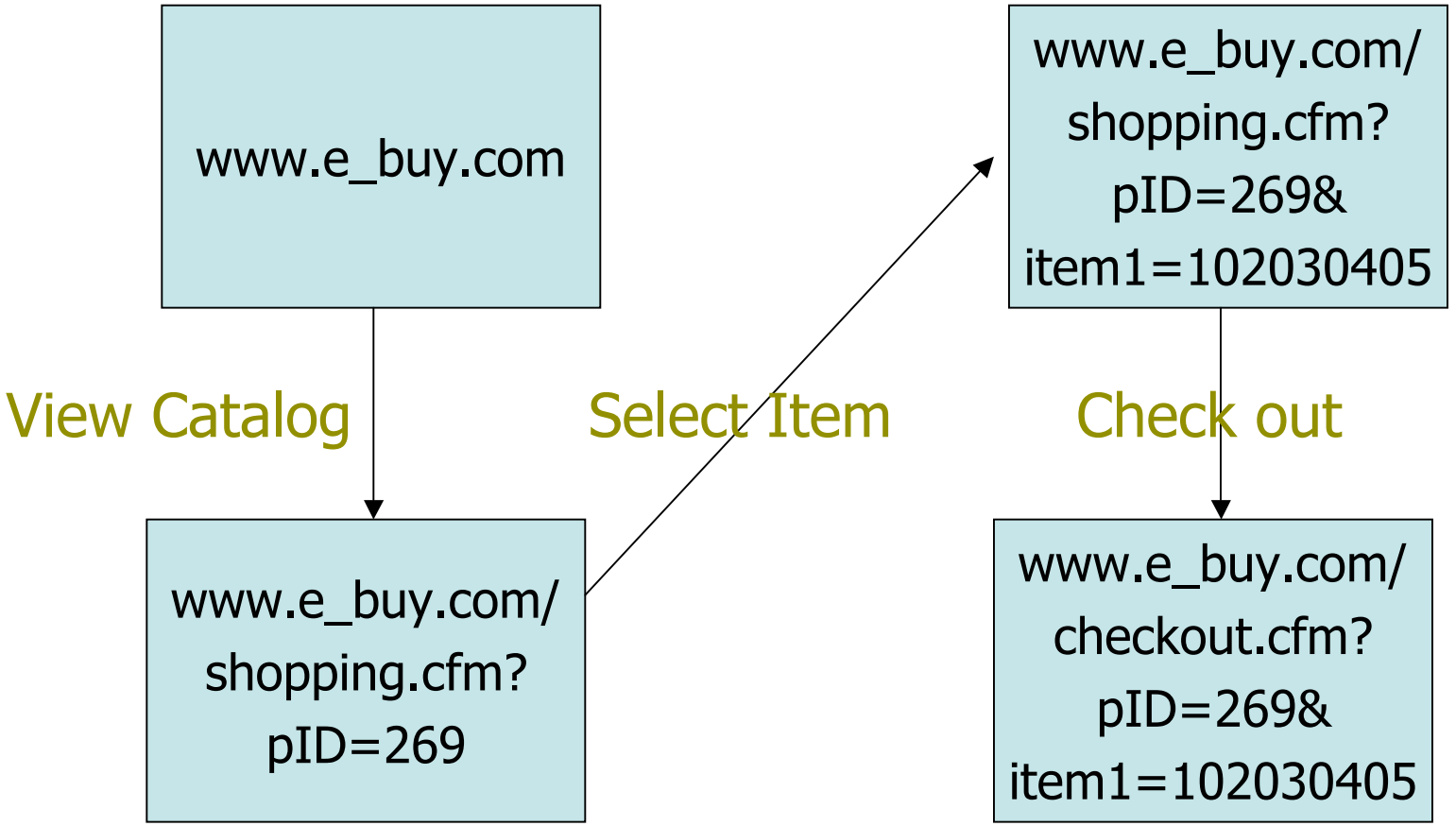
Document object model (DOM)

- Object-oriented interface used to read and write documents
  - web page in HTML is structured data
  - DOM provides representation of this hierarchy

- Examples
  - Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], document.anchors[ ]
  - Methods: document.write(document.referrer)

- Also Browser Object Model (BOM)
  - Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Need for session state



Store session information in URL; Easily read on network

Store info across sessions?

- Cookies
  - A cookie is a file created by an Internet site to store information on your computer

Http is stateless protocol; cookies add state

Cookie

- A named string stored by the browser
  - Accessible as property of the Document object
  - Can be read and written entirely on client side using Javascript
- Accessibility
  - persists for the duration of the browser session (but an expiration date may be given)
  - is associated with the subtree of the document that created it (but a cookie path may be specified)
  - is accessible to pages on the server that created it (but a cookie domain may be declared)

Browser security risks

- Compromise host
  - Write to file system
  - Interfere with other processes in browser environment
- Steal information
  - Read file system
  - Read information associated with other browser processes (e.g., other windows)
  - Fool the user
  - Reveal information through traffic analysis

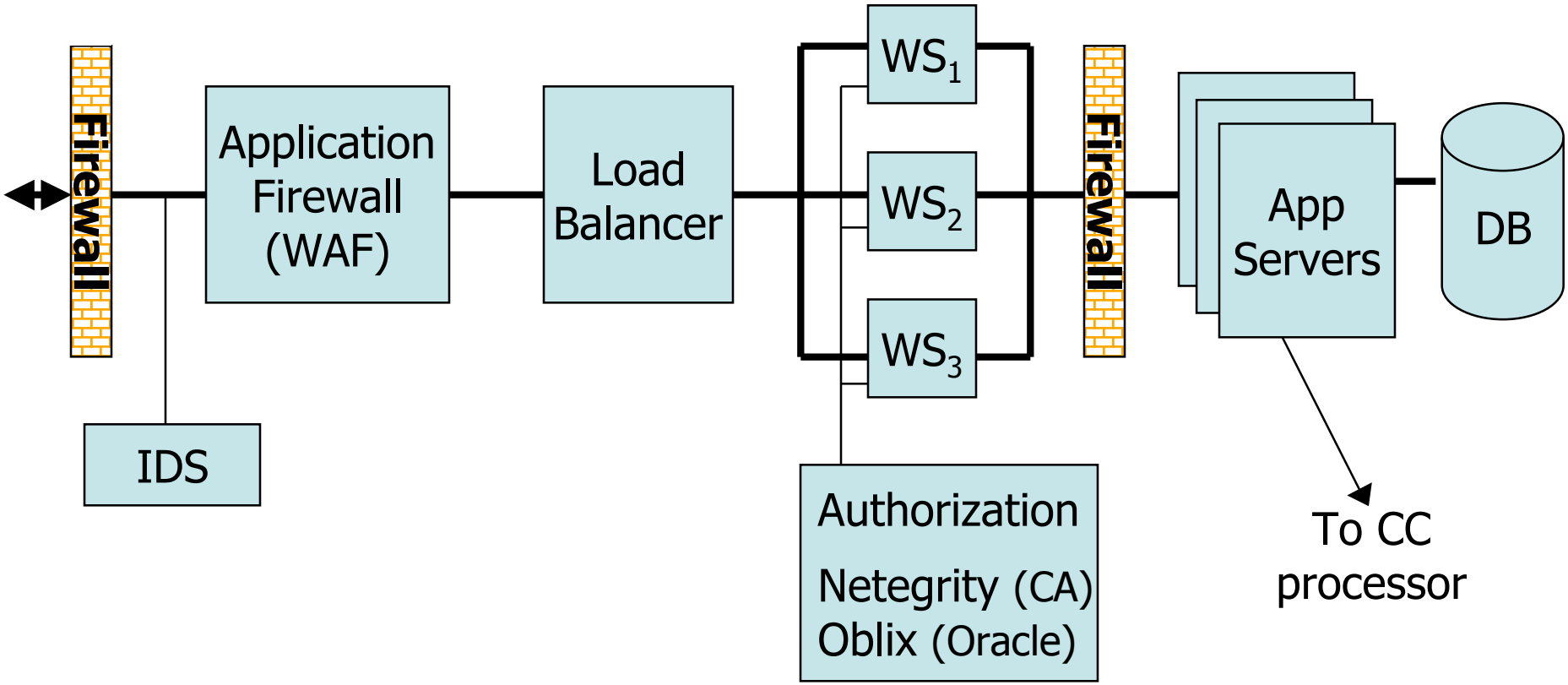
Browser sandbox

- Idea
  - Code executed in browser has only restricted access to OS, network, and browser data structures
- Isolation
  - Similar to OS process isolation, conceptually
  - Browser is a “weak” OS
  - Same-origin principle
    - Browser “process” consists of related pages and the site they come from

Same-Origin Principle

- Basic idea
  - Only the site that stores some information in the browser may later read or modify that information (or depend on it in any way).
- Details
  - What is a “site”?
    - URL, domain, pages from same site … ?
  - What is “information”?
    - cookies, document object, cache, … ?
  - Default only: users can set other policies
    - No way to keep sites from sharing information

Schematic web site architecture



Our focus: web app code

- Common web-site attacks:
  - Denial of Service: ealier in course
  - Attack the web server (IIS, Apache) :
    - e.g. control hijacking: CodeRed, Nimda, …
    - Solutions:
      - Harden web server: stackguard, libsafe, …
      - Worm defense: later in course.
        - » Host based intrusion detection,
        - » Worm signatures generation, shields.
- Today:
  - Common vulnerabilities in web application code

Web app code

- Runs on web server or app server.
  - Takes input from web users (via web server)
  - Interacts with the database and 3rd parties.
  - Prepares results for users (via web server)

- Examples:
  - Shopping carts, home banking, bill pay, tax prep, …
  - New code written for every web site.

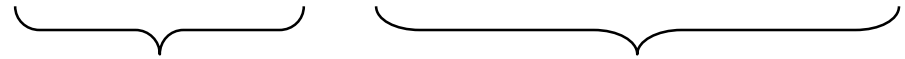
- Written in:
  - C, PHP, Perl, Python, JSP, ASP, …
  - Often written with little consideration for security.

Common vulnerabilities (OWASP)

- Inadequate validation of user input
  - Cross site scripting
  - SQL Injection
  - HTTP Splitting
- Broken session management
  - Can lead to session hijacking and data theft
- Insecure storage
  - Sensitive data stored in the clear.
  - Prime target for theft – e.g. egghead, Verizon.
  - Note: PCI Data Security Standard (Visa, Mastercard)

Warm up: a simple example

- Direct use of user input:
  - http://victim.com/ copy.php ? name=username



  - copy.php:

```
system("cp temp.dat $name.dat")
```

  - Problem:
    - http://victim.com/ copy.php ? name="a ; rm *"

(should be: **`name=a%20;%20rm%20*`**)

Redirects

- EZShopper.com shopping cart (10/2004):

  http://…/cgi-bin/ loadpage.cgi ? page=url

  – Redirects browser to url

- Redirects are common on many sites
  – Used to track when user clicks on external link
  – EZShopper uses redirect to add HTTP headers

- Problem: phishing

  http://victim.com/cgi-bin/loadpage ? page=phisher.com

  – Link to victim.com puts user at phisher.com

  ⇒ Local redirects should ensure target URL is local