CIS 551 / TCOM 401

Computer and Network Security

Spring 2009

Lecture 25

Announcements

- Plan for Today:
  - Web Security Part 2
- Project 4 is due *tonight* at 11:59 pm
- Final exam has been scheduled:
  - Friday, May 8, 2009
  - 9:00am – 11:00am, Moore 216
- Please complete online course evaluations:
  - http://www.upenn.edu/eval

Maintaining State

- HTTP is a stateless protocol
  - Server doesn't store any information about the connections it handles (each request is treated independently)
  - Makes it hard to maintain session information

- Encode state in the URL:
  - …/cgi-bin/nxt?state=-189534fjk
  - Used commonly on message boards, etc. to track thread
- Use HIDDEN input fields
  - When user fills in web forms, the POST request gives server the data
  - You can embed state in invisible "input" fields
- Cookies
  - Store data on the client's machine

Hidden Fields

```
<html>
<head> <title>My Page</title> </head>
<body>
 <form name="myform"
       action="http://…/handle.cgi"
       method="POST">
 <div align="center">
  <input type="text" size="25" value="Name?">
  <input type="hidden" name="Language" value="English">
 <br><br> </div> </form>
</body>
</html>
```

Cookies (Client-side state)

- Server can store cookies on the client machine by issuing:

```
Set-Cookie: NAME=VALUE; [expires=DATE;]
[path=PATH;] [domain=DOMAIN_NAME;]
[secure]
```

- Domain and Path restrict the servers (and paths on those servers) to which the cookie will be sent
- The "secure" flag says that the cookie should only be sent over HTTPS
- Uses:
  - User authentication
  - Personalization
  - User tracking: e.g. Doubleclick (3rd party cookies)

Cookies (cont'd)

- When the client requests a URL from a server, the browser matches the URL against all cookies on the client.
- If they match, then the client request includes the line:

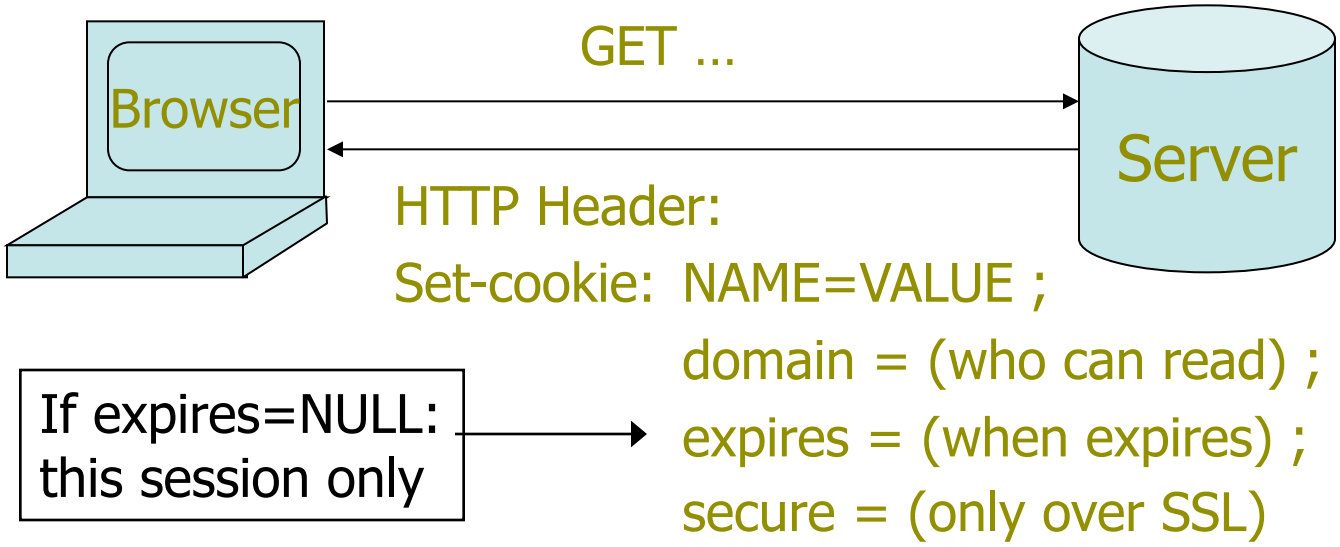
  `Cookie: NAME1=VALUE1; NAME2=VALUE2;…`

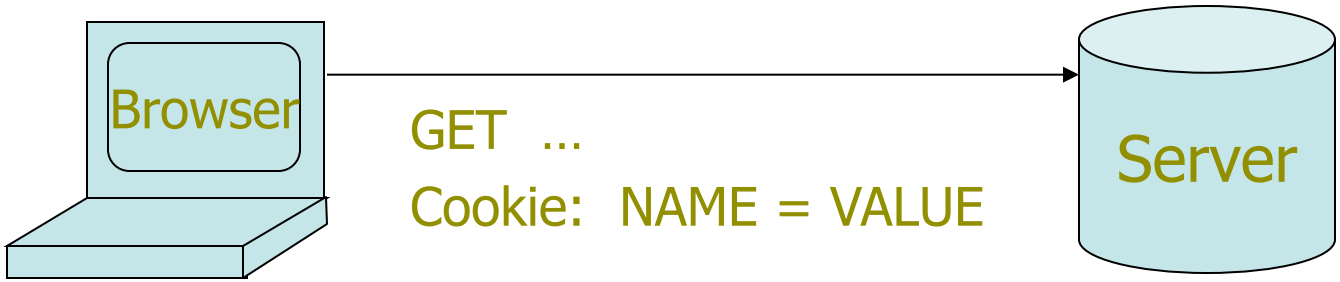
- Notes:
  - New instances of cookies overwrite old ones
  - Clients aren't required to purge expired cookies (though they shouldn't send them)
  - Cookies can be at most 4k, at most 20 per site
  - To delete a cookie, the server can send a cookie with expires set to a past date
  - HTTP proxy servers shouldn't cache Set-cookie headers…

Cookies

Http is stateless protocol; cookies add state

- Used to store state on user's machine





Cookie/Hidden Field Risks

- Danger of storing data on browser:
  - User can change values
- **Silly example**: Shopping cart software.

```
Set-cookie:     shopping-cart-total = 150  ($)
```

  - User edits cookie file (cookie poisoning):

```
Cookie:         shopping-cart-total = 15   ($)
```

  - … bargain shopping.
- Similar behavior with hidden fields:

```
<INPUT TYPE="hidden" NAME=price VALUE="150">
```

Example: dansie.net shopping cart

- http://www.dansie.net/demo.html (April, 2009)

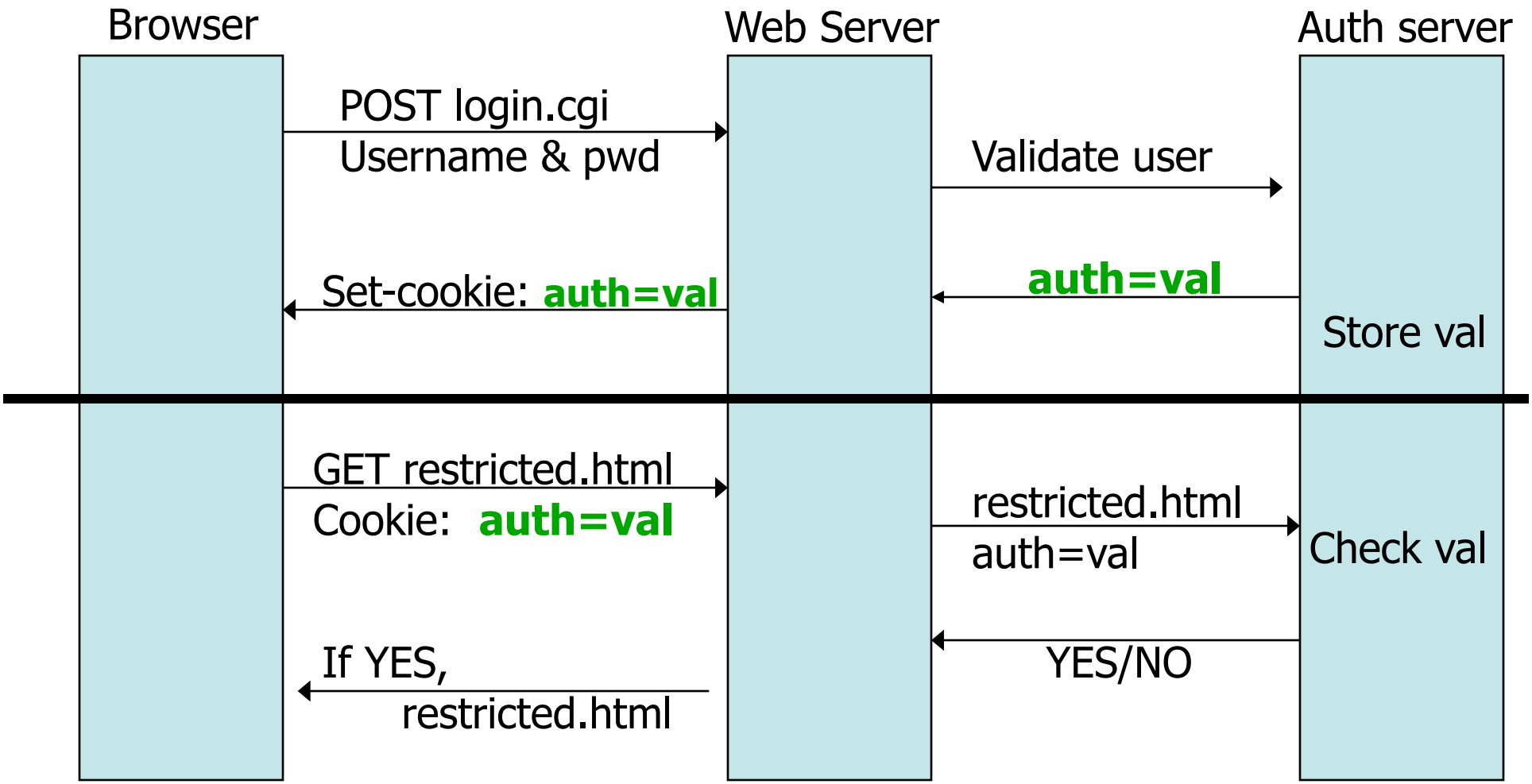
```
<FORM METHOD=POST ACTION="http://www.dansie.net/cgi-bin/scripts/
cart.pl">

<FONT FACE="Times New Roman" COLOR="#000099" SIZE=+1>Black Leather purse
with leather straps<BR>Price: $20.00</FONT><BR>

<INPUT TYPE=HIDDEN NAME=name VALUE="Black leather purse">
<INPUT TYPE=HIDDEN NAME=price VALUE="20.00">
<INPUT TYPE=HIDDEN NAME=sh VALUE="1">
<INPUT TYPE=HIDDEN NAME=img VALUE="purse.jpg">
<INPUT TYPE=HIDDEN NAME=img2 VALUE="purse_large.jpg">
<INPUT TYPE=HIDDEN NAME=return VALUE="http://www.dansie.net/demo.html">
<INPUT TYPE=HIDDEN NAME=custom1 VALUE="Black Leather purse with leather
straps">

<INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">
</FORM>
```

Solution

- When storing state on browser use a Message Authentication Code (MAC) with server's secret key to enforce data integrity.

- .NET 2.0 (probably similar in 3.0):
  - System.Web.Configuration.MachineKey
    - Secret web server key intended for cookie protection
  - HttpCookie cookie = new HttpCookie(name, val);
    HttpCookie encodedCookie =
    HttpSecureCookie.Encode (cookie);
  - **HttpSecureCookie.Decode** (cookie);

Cookie authentication (over https)



Cookie auth is insufficient

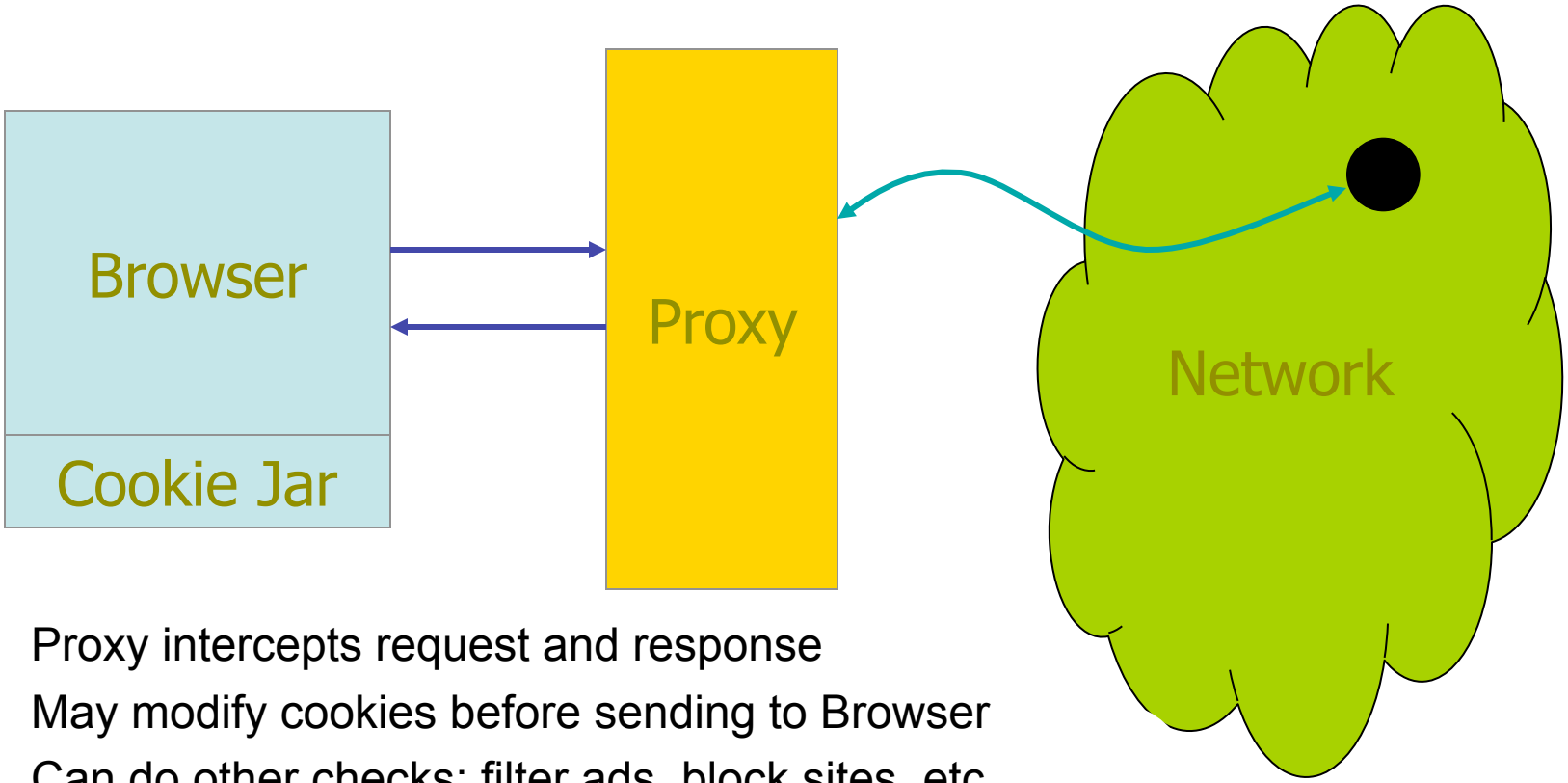
- Example:
  - User logs in to bank.com. Forgets to sign off.
  - Session cookie remains in browser state
  - Then user visits another site containing:

```
<form name=F action=http://bank.com/BillPay.php>
<input name=recipient value=badguy> …
<script> document.F.submit(); </script>
```

  - Browser sends user auth cookie with request
    - Transaction will be fulfilled

- Problem:
  - Cookie auth is insufficient when side effects can happen
  - Correct use: use cookies + hidden fields
  - Hidden fields: store nonces that must be presented to the server
    - Can't be guessed by the malicous web site

Managing cookie policy via proxy



- Proxy intercepts request and response
- May modify cookies before sending to Browser
- Can do other checks: filter ads, block sites, etc.
- This is just a reference monitor for cookies

Sample Proxy:

- Cookie management by policy in *cookiefile*
  - Default: all cookies are silently crunched
  - Options
    - Allow cookies only to/from certain sites
    - Block cookies to browser (but allow to server)
    - Send vanilla wafers instead
- Block URLs matching any pattern in *blockfile*
  - Example: pattern /*.*/ad matches http://nomatterwhere.com/images/advert/g3487.gif

Easy to write your own http proxy; you can try *this* at home

Phishing

- Phishing:
  - Trojan horse e-mails and web sites designed to trick the user into giving up account/pin/password/credit card information.

- December 17, 2007: Gartner Survey
  - Estimated $3.2 BILLION was lost to phishing attacks
  - 3.3% of those surveyed lost money due to phishing
  - (more than in prior years)
  - Most spoofed: PayPal and eBay
  - See:

  www.doshelp.com/scams-fraud/Services/Ebay-Scams.htm

- Goal: Present a plausible experience to the user

Phishing Techniques

- See "Technical Trends in Phishing Attacks"
  - by Jason Milletary (US-CERT)
- Social Engineering
- Bot nets
  - Same infrastructure as Spam mail
- Web site hosting
  - Redirects / obfuscated URLs etc.
- Phishing Kits
  - Pre-generated HTML/e-mail that looks official (graphics, etc.)
- Browser vulnerabilities
  - Borderless popup windows that don't display the address bar
  - Cross-domain vulnerabilities
- XSS using URL redirectors that don't sanitize inputs

Reading browser history

- CSS properties of hyperlinks
- Can also use cache-based techniques:
  - Images and other data in the cache take less time to load, so a script can time how long it takes to load a resource to get some hints about a user's prior browsing.

Violation of the same-origin principle:

"One site cannot use information belonging to another site."

Visited link tracking http://www.safehistory.com/

- Visited links displayed in different color (74% of sites)
  - Information easily accessible by javascript
- Attacks also without javascript

```
<html><head>
<style> a { position:absolute; border:0; } a:link { display:none } </style>
</head><body>
<a href='http://www.bankofamerica.com/'><img src='bankofamerica.gif'></a>
<a href='https://www.wellsfargo.com/'><img src='wellsfargo.gif'></a>
<a href='http://www.usbank.com/'><img src='usbank.gif'></a>
...
</body></html>
```

  - Bank logo images are stacked on top of each other
  - CSS rules cause the un-visited links to vanish
  - Page displays bank logo of site that user has visited

Countermeasures?

- Education and awareness training
- "Security indicators" in the web browser
  - E.g. the yellow address background for https connections in FireFox
- Browser extensions that act as a firewall
  - Can blacklist known phishing sites
- Internet lists of known phishing sites:
  - www.phishtank.com

Do they work?

- Paper: "The Emperor's New Security Indicators: An evaluation of website authentication and the effect of role playing on usability studies" (Schechter *et al.* 2007)
  - Available on the course web pages

- Will customers of an online bank…
  - enter their passwords even if their browsers' HTTPS indicators are missing?
  - enter their passwords even if their site-authentication images are missing?
  - enter their passwords even if they are presented with an IE7 warning page?

Study

- 67 participants:
  - All had accounts at the same bank
  - Mostly Harvard students (not computer scientists/engineers)

- Divided into 3 groups:
  - Group 1: Played a "role" but not told that security was important
  - Group 2: Played a "role" but told that security was important
  - Group 3: Not role playing

- Participants were asked to complete several tasks
  - Check facts about their account balance, last login, last transaction, last statement
- Hints that someone was spoofing:
  - Remove HTTPS indicator
  - Remove site authentication images
  - Present a warning page

Results

	Group 1 Role playing	Group 2 Role playing	Group 3 Per. Acct.
Upon noticing HTTPS missing	0%	0%	0%
Image removed	0%	0%	9%
After Warning	47%	29%	55%
Never (Always logged in)	53%	71%	36%

Main Take-away Ideas (1)

- Security is about Tradeoffs
  - Balance risk vs. expense

- *Principles of Secure System Design:*

- Security is a process
- Least privileges
- Complete Mediation
- System Design
  - Economy of mechanism
  - Open standards
  - Failsafe Defaults

Main Take-away Ideas (2)

- Cryptography is important…
  - Can be used for more than just hiding information
  - Authentication and integrity
- … but not the only facet of security
  - Other risks
  - Social engineering is effective
  - Cryptography applied inappropriately is useless
- So: use it where necessary, and use it correctly
  - See Schneier's book *Applied Cryptography*

Main Take-away Ideas (3)

- Concepts of security:
  - Confidentiality
  - Integrity
  - Availability

- General Mechanisms
  - Authentication
    - Challenge / Response
  - Authorization
    - Reference monitors
    - Access control matrices
  - Audit
    - Logs

Main Take-away Ideas (4)

- Cryptography & Protocol Design
  - Shared vs. Public key cryptography
- Cryptographic protocols can be used for:
  - Authentication, privacy, confidentiality
- Challenge—Response is the fundamental method of authentication
- Nonces, Time stamps, Sequence numbers prevent replay attacks

Main Take-away Ideas (5)

- Malicious Code
  - Viruses & Worms
  - Defense in depth: patching, firewalls, proper configuration, auditing

- Buffer overflows are the #1 vulnerability
  - Choose safe languages:
    - Java, C#, Scheme, ML
  - Be aware of format string and input errors, take care when writing programs and scripts.
  - Software audit and design is important.
  - If you must use C or C++, use StackGuard, ProPolice, or another buffer-overflow preventative measure.

Further study

- Advanced cryptography & cryptographic protocols
  - Elliptic curves
  - Protocol analysis - logic and model checkers
  - Secret sharing, voting
- Systems security
  - Fault tolerance: replication, consensus algorithms

- Additional sources of information (research literature):
  - IEEE Symposium on Security & Privacy ("Oakland conference")
  - Usenix Security conference
  - ACM Conference on Computer and Communications Security
  - Computer Security Foundations Workshop
  - CRYPTO, EUROCRYPT

Final Exam

- Monday, May 8 9:00 - 11:00am Moore 216

- Will cover all the material in the course
  - But will emphasize the new material since Midterm 2

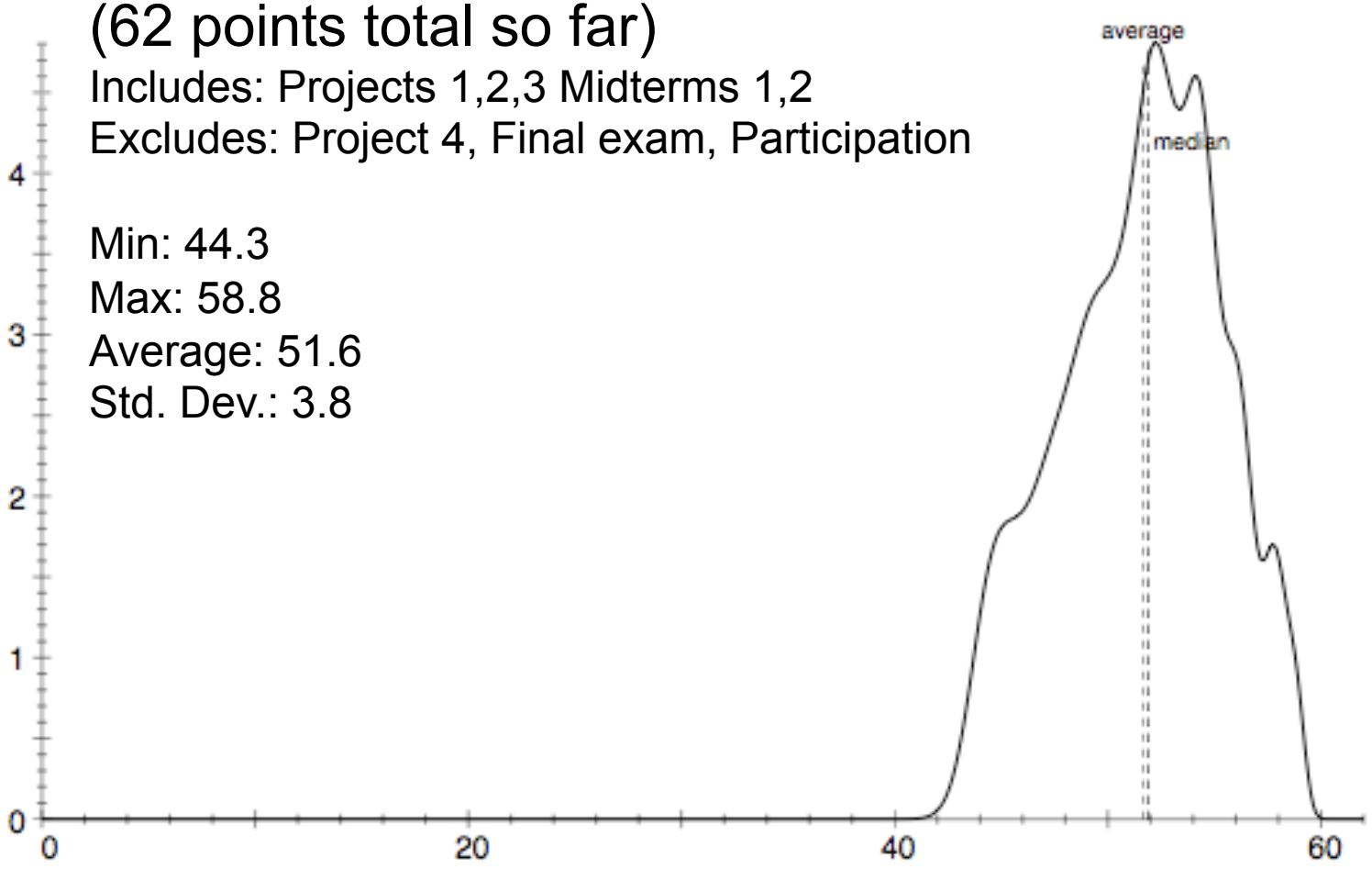
- Format will be similar to previous exams
  - T/F, multiple choice, short answer, short problems
  - The final will have a security analysis/synthesis question

- Send e-mail to make an appointment if you would like to meet with me

Grade Distribution To Date

Weighted grade distribution
(62 points total so far)
Includes: Projects 1,2,3 Midterms 1,2
Excludes: Project 4, Final exam, Participation

Min: 44.3
Max: 58.8
Average: 51.6
Std. Dev.: 3.8



Thanks!