

## User Logins, Profiles, and Personalization

Dr. Michele Weigle

<http://www.cs.odu.edu/~mweigle/CS418-F12/>

## Outline

---

- ▶ Assigned Reading
  - ▶ Chapter 12 "User Logins, Profiles, and Personalization"
- ▶ Simple File Protection
- ▶ Protecting Files with PHP (sessions and cookies)

## Simple File Protection

---

- ▶ Place file `.htaccess` into directory you want to protect
- ▶ Specify

```
AuthType Basic
AuthUserFile /path/to/file/containing/user/credentials
AuthName "MyAuthExampleName"
restrictions
```

*Example coming up*

## `.htaccess`

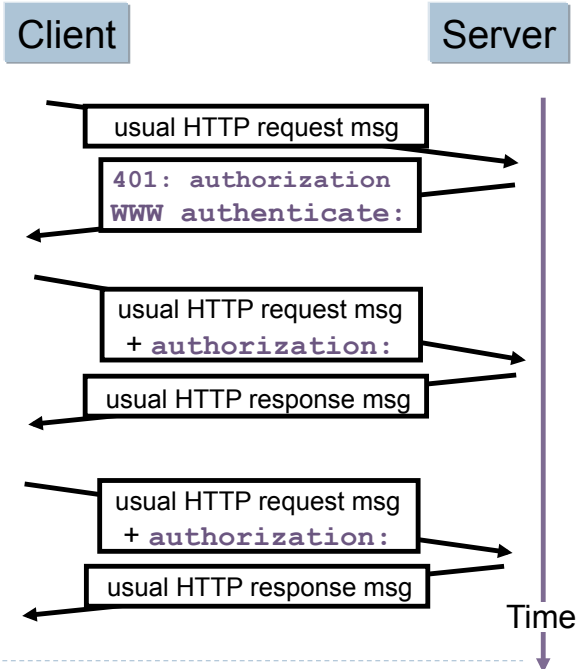
---

- ▶ When you request a page, Apache checks for `.htaccess` in each folder on the path to the requested file
- ▶ Looks for what directory to protect
- ▶ Looks for location of password file
- ▶ No usernames or passwords are sent with initial request, so message sent back to browser requesting login

# HTTP User-Server Interaction

## Authentication

- ▶ HTTP includes a header tag for user to specify name and password (on a GET request)
  - ▶ If no authorization presented, server refuses access, sends WWW authenticate: header line in response
- ▶ Stateless: client must send authorization for each request
  - ▶ A stateless design
  - ▶ (But browser may cache credentials)



## .htaccess Example

<https://mweigle418.cs.odu.edu/~mweigle/textbook/ch12.htm>

```
/home/mweigle/cs418_html/protected/.htaccess:  
AuthType Basic  
AuthUserFile /home/mweigle/passwds/userauth  
AuthName "Restricted"  
<LIMIT GET POST>  
require valid-user  
</LIMIT>
```

Try:

<https://mweigle418.cs.odu.edu/~mweigle/protected>

*Verify with HTTP Live Headers*

## Problems with .htaccess Protection

---

- ▶ Default dialog box is ugly
- ▶ Third-party hosting company may not allow .htaccess
- ▶ Easier to use brute force attacks than with program-driven authorization
  - ▶ sessions, cookies - *coming up next*
- ▶ Not easy to customize
- ▶ But, still use .htaccess to protect non-PHP files (images, PDFs, ZIPs, etc.) or provide an extra level of security when using program-drive auth

## Outline

---

- ▶ Simple File Protection
- ▶ Protecting Files with PHP (sessions and cookies)

# Sessions and Cookies

---

## ▶ Session

- ▶ variable that is kept alive on the server side when someone navigates to a page
- ▶ use this info to track the user throughout the site
- ▶ only exists while user's browser is open

## ▶ Cookie

- ▶ stored on the user's computer
- ▶ less secure than session because user can tamper with cookie
- ▶ exists until it expires
  - ▶ can be used for multiple browsing sessions

# Protect Files – the PHP Way

## Sessions

---

- ▶ Can transport session from page to page
- ▶ Session is destroyed when browser closed
- ▶ Server-side, hence user is NOT able to modify session data
- ▶ `session_start();`
  - ▶ creates a session or resumes the current one
- ▶ associative array `$_SESSION`
  - ▶ `$_SESSION['logged'] = 1;`
- ▶ Testing a session variable:

```
if(isset ($_SESSION['logged']) && $_SESSION['logged'] == 1) {  
    echo "you are logged in";  
} else {  
    echo "you need to login!";  
}
```

# Protect Files – the PHP Way

## Cookies

- ▶ Stored on the user's computer
- ▶ Allows for persistent login, for example
- ▶ Client-side, hence user IS able to modify cookie data
- ▶ `setcookie(name, value, expiration);`
  - ▶ name: used to retrieve cookie
  - ▶ value: value stored in cookie (username, last visit)
  - ▶ expiration: date when cookie will expire/be deleted (if not set, cookie is treated as session cookie – removed at browser restart)
- ▶ associative array `$_COOKIE`
- ▶ Testing a cookie:

```
if($_COOKIE['username'] != "") {  
    echo "your name is: $_COOKIE['username']";  
} else {  
    echo "who are you?";  
}
```

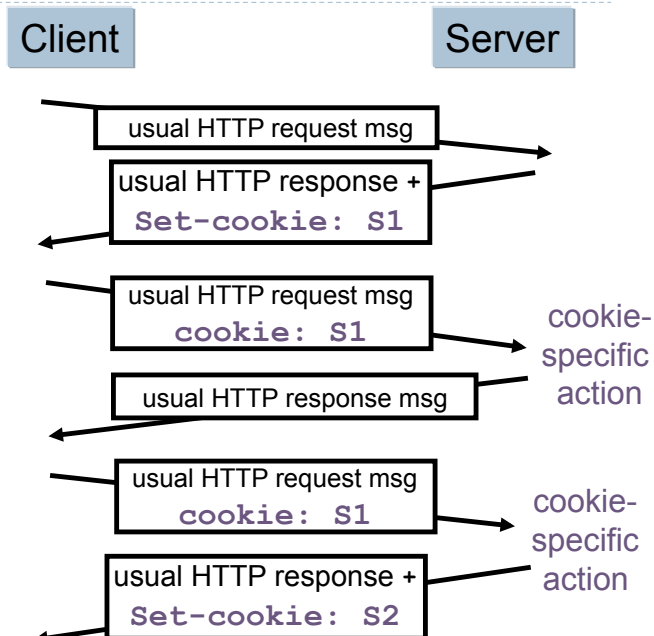
▶ 11

CS 418/518 - Fall 2012

## HTTP User-Server Interaction

### Cookies

- ▶ Server sends "cookie" to browser in response message
  - ▶ `Set-cookie: <value>`
- ▶ Browser presents cookie in later requests to same server
  - ▶ `cookie: <value>`
- ▶ Server matches cookie with server-stored information



▶ 12

CS 418/518 - Fall 2012

## Advantages of Using Sessions and Cookies

---

- ▶ Don't have to worry about passing info about users with form data or through query string
- ▶ All data is stored temporarily on the server
- ▶ Don't have to worry about people trying fake other users through parameters in the address bar
  - ▶ session data is unavailable to users
- ▶ Main difference between sessions and cookies:
  - ▶ amount of time info is available

## Outline

---

- ▶ Simple File Protection
- ▶ Protecting Files with PHP (sessions and cookies)

Up Next:  
Project 2 status reports

# Demo/Walkthrough Time

---

- ▶ Examples from Chapter 12

<https://mweigle418.cs.odu.edu/~mweigle/textbook/ch12.htm>

- ▶ Using PHP for Logins
- ▶ Using Database-Driven Information
- ▶ Authorizing Users to Edit Their Accounts
- ▶ Editing User Accounts
- ▶ Cookie Tracking with PHP
- ▶ Admin Section