CS 312
Internet Concepts

Web Applications:
Internet Search and Webpage Accessibility

Dr. Michele Weigle
Department of Computer Science
Old Dominion University
mweigle@cs.odu.edu

http://www.cs.odu.edu/~mweigle/CS312-F10/

Outline

- Generic Searching
- How Google Works
- Digital Preservation
  - » Internet Archive
- Webpage Accessibility
  - » make your webpages more popular
  - » keep your webpages private

Generic Search Engines

- Some very popular ones
  - » Google
  - » Bing
  - » Yahoo!
- Some common characteristics
  - » Huge set of results
  - » Extremely prompt retrieval
  - » Organized retrieval results

How do they achieve such amazing results?

How Do Search Engines Work?

- Prep work
  - » *crawling* – an automated way of browsing the web for pages
  - » archiving, analyzing, organizing, indexing
- Retrieval
  - » query matching
  - » ranking search results
  - » displaying search results

Search Query Syntax

- The space between keywords is interpreted as AND for some search engines, but as OR for others.
- Query: *I'm interested in cancer in adults.*
  - » Boolean logic: AND
  - » Search query: +cancer +adults
- Query: *I'm interested in radiation, but not nuclear.*
  - » Boolean logic: NOT
  - » Search query: radiation -nuclear

Outline

- Generic Searching
- How Google Works
- Digital Preservation
  - » Internet Archive
- Webpage Accessibility
  - » make your webpages more popular
  - » keep your webpages private

Google's Three Main Components

- Crawling
- Indexing
- Page Weighting

http://ppcblog.com/how-google-works/

Google's Crawler, the Googlebot

- Consists of many computers requesting and fetching pages
  - » Googlebot can request thousands of different pages simultaneously
- When Googlebot fetches a page, it adds all of the links in the page to a queue for subsequent crawling (*deep crawling*)
  - » Googlebot can quickly build a list of links that can cover broad reaches of the web
- Google continuously recrawls popular frequently changing web pages at a rate roughly proportional to how often the pages change (*fresh crawls*)

Submitting URLs to Googlebot

- http://www.google.com/addurl.html
- Rejects URLs Google suspects are trying to deceive users
  - » including hidden text or links on a page
  - » stuffing a page with irrelevant words
  - » cloaking (aka bait and switch)
  - » using sneaky redirects
  - » creating doorways, domains, or sub-domains with substantially similar content
  - » sending automated queries to Google
  - » linking to bad neighbors.

Google's Indexer

- Parses webpages discovered by the Googlebot
- Converts them into a set of hits (word occurrences).
  - » For each document in which the word appears, indexer stores the position of the word, font size, capitalization
- Creates an *inverted index* that is indexed by word pointing to documents

Forward Index

Document	Words
Document 1	the,cow,says,moo
Document 2	the,cat,and,the,hat
Document 3	the,dish,ran,away,with,the,spoon

Inverted Index

Word	Documents
the	Document 1, Document 3, Document 4, Document 5
cow	Document 2, Document 3, Document 4
says	Document 5
moo	Document 7

Google's Page Weighter, *PageRank*

- Assigns each webpage a relevancy score
- A webpage's PageRank depends upon
  - » frequency and location of keywords within the page
  - » how long the webpage has existed
  - » number of other webpages that link to the page
- Most important factor is number of pages that link to the page
  - » essentially, PageRank is a “vote”, by all the other pages on the Web, about how important a page is

http://computer.howstuffworks.com/google1.htm

Manipulating PageRank / Webspam

- Google bomb
  - » is created if many sites link to the page using the same anchor text (increases the page rank)
  - » ex: In 1999, a search for "more evil than Satan himself" resulted in the Microsoft homepage
- spamdexing
  - » deliberately modifying HTML pages to increase the chance of their being placed close to the beginning of search engine results
- link doping
  - » embedding a large number of gratuitous hyperlinks on a website, in exchange for reciprocal links
- Google Jacking (page hijacking)
  - » creating a rogue copy of a popular website which shows contents similar to the original to a web crawler, but redirects web surfers to unrelated or malicious websites.

Outline

- Generic Searching
- How Google Works
- Digital Preservation
  - » Internet Archive
- Webpage Accessibility
  - » make your webpages more popular
  - » keep your webpages private

Digital Preservation

- Much of the record of our lives is digital.
  - » How do we keep these records?
  - » How do we "walk down memory lane"?
- Backup strategies
- Recording and archiving the web
- Accessing past versions of webpages

Web Science and Digital Libraries Research Group at ODU (http://ws-dl.blogspot.com/)

Internet Archive

- Internet Archive
  - » non-profit that was founded to build an Internet library, with the purpose of offering permanent access for researchers, historians, and scholars to historical collections that exist in digital format
  - » http://www.archive.org
- Internet Archive's Wayback Machine
  - » allows you to browse through 85 billion web pages archived from 1996 to a few months ago
  - » http://www.archive.org/web/web.php

Memento

- Time Travel for the Web
  - » http://www.mementoweb.org/
- MementoFox – Firefox add-on
  - » https://addons.mozilla.org/en-US/firefox/addon/100298/
- Uses Internet Archive and others to present a webpage as it existed at a certain date/time

Web Science and Digital Libraries Research Group at ODU (http://ws-dl.blogspot.com/)

Outline

- Generic Searching
- How Google Works
- Digital Preservation
  - » Internet Archive
- Webpage Accessibility
  - » make your webpages more popular
  - » keep your webpages private

Web Page Accessibility

- Are all pages posted on the web necessarily available to all?
- What about dynamically generated pages?
- Are there some ways to help enhance accessibility?
- Is it difficult to be able to restrict access on certain pages?

Enhancing Your Page's Popularity

- Submit your page to search engines
  - » http://www.google.com/addurl.html
- 12 Things to Do to Improve Your PageRank
  - » http://stason.org/articles/money/seo/google/12_things_to_do_to_improve_your_site_google_page_rank.html
- 12 Things **Not** to Do to Improve Your PageRank
  - » http://stason.org/articles/money/seo/google/12_things_not_to_do_to_improve_your_site_google_page_rank.html

Restricting Accessibility

- Robots Exclusion Protocol
  - » used to give instructions about a site to web robots (like Googlebot)
  - » instructions are in placed in `/robots.txt` at the root of a website
    - ❖ need to be the website owner to use this
- Robots can ignore your `/robots.txt`
  - » malware robots - scan the web for security vulnerabilities
  - » email address harvesters used by spammers

http://www.robotstxt.org/

Examples of /robots.txt

Against indexing the entire site

```
User-agent: *
Disallow: /
```

*User-agent: **
means all unspecified robots.

Against indexing selected folders or files

```
User-agent: *
Disallow: /~mweigle/
Disallow: /~mweigle/files/foo.txt
```


Examples of /robots.txt

To allow only specific robots

```
User-agent: googlebot
Disallow:
User-agent: yahoobot
Disallow:
User-agent: *
Disallow: /
```

Does not disallow anything for googlebot or yahoobot.

Disallows all to all other robots.

What Can a Webpage Author Do?

- Use robots <meta> HTML tag at the beginning of your webpage
  - » prevent individual pages from indexing

<meta name="robots" content="*tag1, tag2*">

- Possible *tags*
  - » noindex – don't index this page
  - » nofollow – don't scan this page for links to crawl
- Default is index, follow

http://www.robotstxt.org/

Example robots meta tags

Add in the <head> section one of the tags :

```
<meta name="robots" content="index, follow">
<meta name="robots" content="index, nofollow">
<meta name="robots" content="noindex, follow">
<meta name="robots" content="noindex, nofollow">
```

Accessibility with Robot Specifications

- Will all robots follow the instructions?
  - » not the bad guys
- Can pages still can be accessed when not indexed?
  - » if they're not protected
- Is there any way available to prevent public access but still allow protected access?
  - » yes!

To Prevent Public Access

- Basic authentication
  - » allows web browsers to provide credentials in the form of a user name and password when making a request
  - » protection is per-directory, not per-file
  - » the requesting client must authenticate itself to the server with a user-ID and a password for each directory being protected
- Assumes that the connection between the client and server computers is secure and trusted
  - » all information sent in plain-text
- Could use HTTPS for encryption

Basic Authentication and .htaccess

- Special file named `.htaccess`
  - » must be publicly readable (chmod 644)
  - » '.' is part of the filename
    - ❖ to list files that have a '.', use the 'ls –a' command in Unix
- States where a corresponding encrypted file (generated in some special way) for this authentication is located
- Only recommended if you don't have access to main server configuration files
  - » i.e., you don't own the webserver

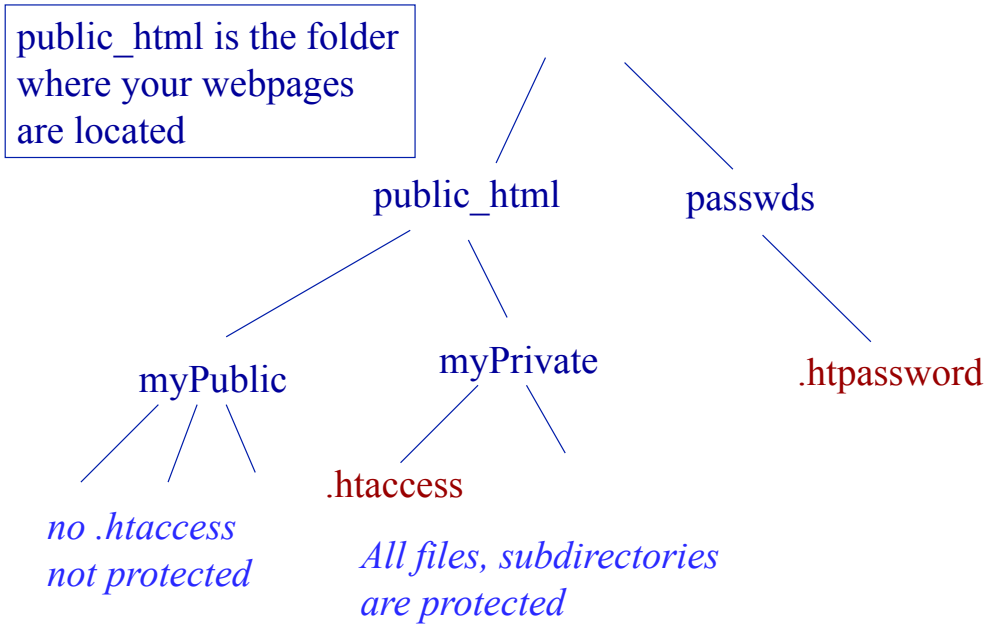
.htaccess Operation

- When a HTTP server tries to access a directory, it first looks in the directory for this file
- If this file does not exist at the directory
  - » the server considers that this directory is not protected and accesses the files in it normally.
- If this file exists at the directory
  - » the server reads this file to find out the location of the encrypted file
- Server reads the encrypted file and uses the information there to verify authentication

.htaccess Operation

- Before accessing any files in this directory, the server checks the request's authentication header.
- When the authentication is satisfied by the request information, then the server delivers the requested file in this directory to the requesting client.
- If not satisfied, the server sends code 401 (unauthorized)
  - » we'll look more at HTTP codes later

Example Directory Structure

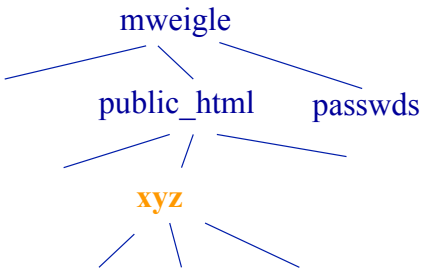


Example .htaccess File

```
AuthType Basic
AuthName "Password Required"
AuthUserFile /home/mweigle/passwds/.htpassword
```


Example

- We want to protect all files in the directory
  `/home/mweigle/public_html/xyz/`
- Only allow user **John** with password **x123y**



Example
Generate the Encrypted File

- Run the encryption program `htpasswd` (no dot in the front) as a Unix command

  /usr/apache/bin/htpasswd -c ~/passwds/.htpasswdxyz John
  - » the parameter -c means to create a new encrypted file.
  - » .htpasswdxyz is the name of the file to be generated
  - » John is the username

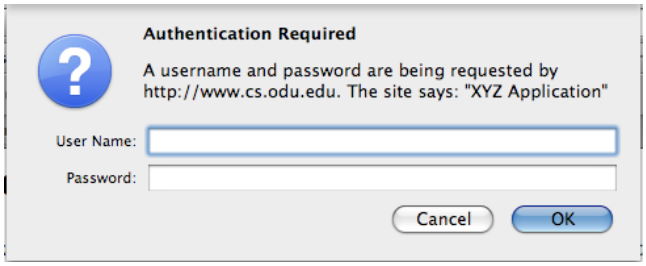
- The program will ask for the password twice

```
% /usr/apache/bin/htpasswd -c ~/passwds/.htpasswdxyz John
New password:
Re-type new password:
Adding password for user John
```


Example
Generate the .htaccess File

- Create a Unix text file named .htaccess, stored in ~/public_html/xyz/

```
AuthType Basic
AuthUserFile /home/mweigle/passwds/.htpasswdxyz
AuthName "XYZ Application"
Require valid-user
```



http://www.cs.odu.edu/~mweigle/xyz/

Example
File Permissions

- ~/public_html/xyz/
  - » chmod 755
  - » folder is accessible and readable by everyone
- ~/public_html/xyz/.htaccess
  - » chmod 644
  - » file is readable by everyone
- ~/passwds/
  - » chmod 755
  - » folder is accessible and readable by everyone
- ~/passwds/.htpasswdxyz
  - » chmod 644
  - » file is readable by everyone

More About .htaccess

- Require
  - » allow any valid user (in the encrypted file)

    Require valid-user
  - » specify which users can have access

    Require user John

More info:
http://httpd.apache.org/docs/2.0/howto/auth.html#basic

.htaccess and Access Control

- Access Control (apart from authentication)
  - » control who can and can't view the page
  - » use 'Allow' and 'Deny' directives
    - Allow from *{hostname, IP address, domain name}*
    - Deny from *{hostname, IP address, domain name,* all*}*
  - » use 'Order' directive to specify the order that they should be applied (usually deny before allow)

```
Order Deny,Allow
Deny from all
Allow from .cs.odu.edu
```

only allows browsers from .cs.odu.edu domain to view the page or authenticate

http://www.cs.odu.edu/~mweigle/xyz/

Safety of Basic Authentication

- Only as good as the username and password
- If not using HTTPS, usernames and passwords are sent in plain-text

Outline

- Generic Searching
- How Google Works
- Digital Preservation
  - » Internet Archive
- Webpage Accessibility
  - » make your webpages more popular
  - » keep your webpages private