Web Server Admin

Don Porter

Web Serving

- Static Content
- Dynamic Content and Security
- Distribution and Scaling

HTTP

- HyperText Transport Protocol
  - Usually implemented on TCP port 80
- Simple, stateless
  - Most conversations are: connect, get something, close connection
- You can actually telnet to a web server and type HTTP commands!

$ telnet www.oscar.cs.stonybrook.edu 80

GET /

Apache Web Server

- Lots of good web servers out there: IIS, lighttpd, etc.
  - Apache is a very popular one, though.
  - Most people use version 2
- Fairly simple to install
- Can be configured to run multiple sites on the same server

Simple Example Apache http.conf

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

<VirtualHost *:80>
DocumentRoot /www/example1
ServerName www.example.com

# Other directives here

</VirtualHost>


<VirtualHost *:80>
DocumentRoot /www/example2
ServerName www.example.org

# Other directives here

</VirtualHost>
```

Key Points

- Easy to host multiple sites:
  - Just define a new virtual hosts
- DocumentRoot:
  - Just a directory full of html, javascript, css, etc.
  - index.html is the default page if none specified

Other useful directives

- ErrorLog: specify the file for error messages
  - Useful for debugging, can be different per VirtualHost
- ServerAdmin: specify the email address of the site admin (you)

.htaccess

- Site owners can specify access control on files using a file called .htaccess (in the same dir with content)
- Example (only allow access from SBU):

```
<Limit GET>
order deny, allow
deny from all
allow from .sunysb.edu
allow from .stonybrook.edu
allow from 129.49.
allow from 130.245.
</Limit>
```

Dynamic Content

- Most of the web isn't just simple web pages anymore
- You can think of most pages as interactive applications
  - E.g., Facebook shows different pages to different users

Idea: Server-side Apps

- Rather than just return the contents of an html file
- Run a program that outputs html, return that
- Similar to a Unix pipe

Common Gateway Interface (CGI)

- Basically, just a standard for how to pass input/ output between a web server and an application
- CGI applications can be implemented in any language
  - Perl, PHP, Python, Ruby are popular

Storing Data

- Suppose I want to store data input by the user
  - E.g., a Facebook-style wall of status posts
- I could put these in a file on the server
- But often easier to use a database
  - Decouples data management from deploying scripts

LAMP Stack

- Linux Apache MySQL Perl/PHP/Python

- MySQL is a popular, open-source database
- Many Linux server installation discs make it easy to bring up a LAMP stack quickly

Script Security

- When you run a script on your server, that script is (generally) just like any other program on your server
- A compromised script can compromise your server
- Proceed with caution
- A lot of script security amounts to being resilient to carefully crafted, malicious inputs
- And limiting the damage a bad script can do

Input Example

- Suppose I am a script that stores user profiles as <name>.txt
- I have a form on my webpage that asks for a name:

```
name = form_input("name");
profile = execute("cat " + name + ".txt");
print profile;
```

- Any issues?

Code injection

- What if I type in as my name:

  Insecure.txt; rm *; echo Hahahaha> pwned

profile = execute("cat " + name);

= execute("cat Insecure.txt; rm *;
echo Hahaha > pwned.txt")

How to deal with inputs?

- In general, you have to carefully check that they are what you expect
  - No escape characters or other unusual strings
- Perl has something called “Taint mode”
- Basically, scripts that use unchecked input as output to a command will be killed
  - Developer still has to write good checks

Sandboxing

- In order to limit the damage of a bad script, you may also want to run it in a sandbox
- BSD has something called a jail: limited view of the file system, limited access to system resources
- Other sandboxing tools exist for this purpos

JavaScript

- In addition to running code on a web server to generate content, you can also load code into a user's browser
  - Service some clicks locally
- JavaScript is the main language for client-side coding

AJAX and Web 2.0

- Most recent web apps use a combination of Javascript on the client and a server-side application
- Server-Side app may not generate whole pages, but may handle smaller requests
  - XML is often used to exchange data between Javascript and the server
  - Hence the name: Asynchronous Javascript And XML

Scaling Up

- As we've discussed before, round-robin DNS lets you have multiple web servers
- And running a database on the back-end lets you have multiple front-end CGI scripts
- You can also run more web servers in the cloud

Content Distribution Networks (CDN)

- Big content providers can improve latency and reduce their own bandwidth by placing data close to users
  - E.g., netflix, facebook, etc.
  - Read-only data
  - Why should NY users have to pull all of their data from CA?
- Companies like Akamai transparently redirect you to the geographically closest server for your content
- Pretty expensive option, mostly for bigger ventures

Proxying, Caching, and Filtering

- Many organizations only allow outgoing web traffic through a proxy server
- 2 main reasons:
  - Caching: Can store (static, public) pages and serve locally, saving bandwidth
  - Filtering: Can refuse to let you see facebook, or illicit pages