

CGI Programming

Mechanisms
Techniques
Resources

This Lecture

- Provide description and structure of CGI on Unix Apache systems
- Describe security, permissions, and cgiwrap
- Explain execution environment and debugging techniques
- Introduce Perl and CGI module
- CGI and SSI

CGI Defined

- Program with a URL
- Input comes embedded in URL or in standard input (stdin)
- Its output delivered as a web page
- Web server executes program
 - Program uses web server CPU
 - Program sees web server filesystems
 - Web server initiates network connections, etc

Requirements for CGI

1. Executable program
2. Residing in executable location
3. Output is content type and content
4. **NO ERROR** output (stderr)

Simplest CGI

```
#!/bin/sh
echo "content-type: text/plain"
echo ""
echo "Hello, world."
```

1. Program is shell script
2. Lives in paco's cgi-bin
3. Prints page type and content
4. No error output



1. Executable program

- Anything that can be executed:
 - Script (shell, perl, tk/tcl, python, etc)
 - Compiled binary (C, C++, Fortran, etc)
- Must write to stdout
- Cannot interact with stdin
- Cannot write stderr

2. Location of program

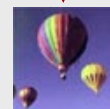
- `/home/USER/public_html/cgi-bin`
- `/cgi-bin` (if you're the webmaster)
- Determines URL
 - Can have subdirectories of cgi-bin
 - Determines program's working directory
- Web server **prevents** other locations



3. Output

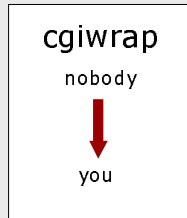
- Web uses MIME like e-mail
- Text string indicates content type
- Content follows immediately inline

```
Content-type: image / jpeg
1020A8BE008AF8AB7C787E
DF8F88A898009FE897B7C9
...
```



Next Topics

- CGI-wrap is your friend
- Dealing with input
- Debugging information
- Perl CGI



cgiwrap - security for CGI

- Alters URL of CGIs
- Executes programs with your userid
- Protects your files, and gives CGIs access to your files

<http://www.cs.virginia.edu/cgi-bin/cgiwrap/userid/script>

1. Web server
2. Global cgi-bin with cgiwrap in it
3. Your userid
4. Name of a script in your ~/public_html/cgi-bin

cgiwrap - Caveats

- Standard environment not set
- Non-standard environment is set
- Current directory is not obvious
- cgiwrap is specific to UVA. Some other sites use it, but not many

Set These Variables!

PATH
LD_LIBRARY_PATH

<http://www.cs.virginia.edu/cgi-bin/cgiwrap/paco/test-cgi>

Dealing with Input

- **GET** query parameters are in URL and must be parsed by delimiters
- **POST** query parameters come in on stdin
- Use a CGI library and you don't care!

GET

.../cgiwrap/paco/getcgi?name=Paco&id=12&search=y

POST

POST /cgi-bin/cgiwrap/...

name=Paco

id=12

search=y

Input

- Query parameters
- Form fields
 - See HTML docs or use FrontPage to design forms



```
<html><head><title>cool</title></head><body>
<form action="/cgi-bin/cgiwrap/paco/test.cgi" method="GET">
Paco is<br>
<input type="radio" name="example" value="cool" >cool <br>
<input type="radio" name="example" value="smart" > smart <br>
<input type="radio" name="example" value="boring" >boring<br>
<input type="submit" value="Tell Him">
</form></body></html>
```

Quick Forms Example

- Selectors
- Check boxes
- Radio buttons
- Free-form text
- Multi-select lists



QUERY_STRING=STATE=VA&GENDER=male&DAY=Monday&DAY=Wednesday&COMMENTS=Comments...

How You See Input

- Variables with values
- Radio buttons restrict choices
- Selectors offer a list
- Text accepts free-form strings (**security!!!**)

Examples

```
FirstName=Paco
AreaCode=804
Comments="I like
this lecture..."
Hacker=
"; cd /; rm -rf *;"
```

Debugging: cgiwrap



- **cgiwrapd**: debugging version of cgiwrap
- Executes CGI
- Shows report of output, including stderr
- Use "cgiwrapd" in URL instead of "cgiwrap"

Perl in CGI: Quick Summary

```
#!/usr/bin/perl
use CGI qw( :all );

$f = new CGI;
if( $f->param() ){
    $f->import_names( 'Q' );
}

print $f->header;
print $f->start_html();
print $Q::name;
print $f->dump;
print $f->footer;
```

- Path to **interpreter**
- Use **CGI** module
- Create **CGI** object
- Look at **query parameters**
- Create **namespace**
- Return **content**

Perl's CGI module

```
$f = new CGI;
$f->textfield(
    -name => 'city',
    -size => 20 );
$f->param( 'city' );
$f->import_names( 'Q' );
if( defined $Q::city )...
my $url = $f->url();
```

- Treats CGI query as an object
- Object has many methods
- Creates namespace for query parameters
- Handles GET/POST implicitly

How I do Perl CGI

```
&printHead;

if( $f->param() )
{
    &checkSubmission();
    if( $errString ne "OK" )
    {
        &errMsg;
        &printForm;
    } else {
        &handleSubmission;
    }
} else {
    &printForm;
}

&printTail;
```

- Define a few major functions
- Use CGI object to print form fields
- Use nice error handling
- Main loop is very simple

SSI and CGI



- Allows formatting of web page with nice tools (e.g. FrontPage)
- Allows flexible dynamic content

SSI and CGI Explained

- Use include virtual directive
- Put formatting commands around the CGI's output

```
...  
<h1>English Town</h1>  
<table bgcolor="green">  
<!-- #include virtual="/cgi-bin/cgiwrap/paco/englishTown" -->  
</table>  
...
```

- CGI is re-executed each time the page is loaded.