HTTP Use Cases

Ivan Pepelnjak (ip@ipSpace.net)

ipSpace.net AG

Generating Binary Files in Scripts

Challenge

Generate binary content in a script and send it to browser client

Use cases:

- Images – watermarking, graphs ...
- PDF – report generation, watermarking ...
- Downloads with application-specific authentication

Solution

1. Trap errors, generate regular HTML on errors
2. Generate binary content

Mandatory:

2. Set **content-type** headers

Optional

3. Set **content-length** header
4. Set **attachment** header (desired browser-side filename)

5. Set STDOUT to binary mode (language-specific)
6. Write the content to STDOUT
7. Exit immediately

Example: Send PDF File To Client

```
# CGI object in $q
# PDF content in $pdf
# File name in $dfname
#
 print $q->header (
    -type=>'application/pdf',
    -attachment=> basename($dfname),
    -expires=>'+1h');
 binmode STDOUT;
 print $pdf;
```

Yeah, it's in PERL. Get used to it ;)

Detect Content Linking

Challenge

People like to link to images, videos ... hosted by others

- I get the eyeballs, you get the traffic
- Important if you pay for the traffic and/or have quotas

We want to detect cross-site linking

- Display “content stolen” images
- Redirect to watermarking scripts

Referer Header

- Set by browsers when generating HTTP requests
- Contains URL of
  - ➔ Page elements: HTML page that triggered the request
  - ➔ HTML page on which the user clicked the link to this page
- Missing when
  - ➔ User enters URL in the address bar
  - ➔ Switching from HTTPS to HTTP
  - ➔ On cross-domain requests (browser-dependent)
- Easily spoofed by
  - ➔ scripts using client-side HTTP (ex: LWP, curl)
  - ➔ JavaScript with XMLHttpRequest

Summary: It is not a reliable authentication method

Using Referer with Apache mod_rewrite

```
RewriteCond %{HTTP_REFERER}!^http://(www\.)?example\.com [NC]
RewriteCond %(REQUEST_FILENAME).jpg$ [NC]
RewriteRule - [F,L]

RewriteCond %{HTTP_REFERER} ^$ [NC]
RewriteCond %(REQUEST_FILENAME).jpg$ [NC]
RewriteRule - [F,L]
```

Hints

- [NC] in RewriteCont = Case insensitive
- [F] in RewriteRule = Forbidden (403)
- [L] in RewriteRule = Last rule (stop rule matching)

Application-Level Session State

Challenge

- HTTP is stateless (no information is kept across requests)
- Applications have to remember per-user state (or variables)

Examples

- Shopping cart
- User settings

Solutions

Browser-side solutions

- Local storage and appCache (newer browsers only)
- HTTP Cookies

Server-side solutions

- Store user data in cookie or
- Use session manager and save session ID in cookie

Session managers:

- Embedded in most scripting languages (ex: PHP, ASP.NET)
- Use worker process memory (ASP), local files (PHP) or other storage mechanisms (database, key-value caches ...)

Advanced details in *scale-out architectures*

HTTP Cookies

Storage

- Per web site (= host name from URL) or per domain
- Transient (deleted after browser closes) or permanent

Set by:

- Client-side JavaScript
- With **set-cookie** header in HTTP responses

Used by

- Client-side JavaScript (exception: HttpOnly cookies)
- Server-side scripts (usually available in special data structure)

Setting Cookies

```
GET / HTTP/1.1
Host: www.google.si
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:9.0.1) Gecko/20100101 Firefox/9.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive

HTTP/1.1 200 OK
Date: Sun, 26 Feb 2012 17:53:13 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie: PREF=ID=fd6d6dc3d5dee125:FF=0:TM=1330278793:LM=1330278793:S=txJW78v2aQ9Efa20;
expires=Tue, 25-Feb-2014 17:53:13 GMT; path=/; domain=.google.si
Set-Cookie: NID=57=d5LOsrShr6TIih_0BDm7x4_eIECSCthCIqfYTVPPHQnep1C86K7SFx8dxQry4rHFZ6yV3oH-
fPNdm3zeygU0Ch8UgAF8Ppz58i6MtaCf-XGfIC-u9ki7tbJfIicH9dUx; expires=Mon, 27-Aug-2012 17:53:13
GMT; path=/; domain=.google.si; HttpOnly
Content-Encoding: gzip
Server: gws
Content-Length: 17326
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
```

Sending Cookies

```
GET /images/nav_logo103.png HTTP/1.1
Host: www.google.si
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:9.0.1) Gecko/20100101 Firefox/9.0.1
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Referer: http://www.google.si/
Cookie: PREF=ID=fd6d6dc3d5dee125:FF=0:TM=1330278793:LM=1330278793:S=txJW78v2aQ9Efa20;
NID=57=d5LOsrShr6TIih_0BDm7x4_eIECSCthCIqfYTVPPHQnep1C86K7SFx8dxQry4rHFZ6yV3oH-
fPNdm3zeygU0Ch8UgAF8Ppz58i6MtaCf-XGfIC-u9ki7tbJfIicH9dUx

HTTP/1.1 200 OK
Content-Type: image/png
Last-Modified: Fri, 17 Feb 2012 20:52:13 GMT
Date: Sun, 26 Feb 2012 17:53:13 GMT
Expires: Sun, 26 Feb 2012 17:53:13 GMT
Cache-Control: private, max-age=31536000
X-Content-Type-Options: nosniff
Server: sffe
Content-Length: 29576
X-XSS-Protection: 1; mode=block
```

Typical Session Manager Behavior

Is session cookie set?

- No – new session
- Yes: can we read session data?
  - No – new session
  - Yes - read session data from session store

New session

- Clear session data
- Create new entry in session store (file ...)
- Set session cookie in response header

Script cleanup behavior

- Store session data in session store

User Authentication

Challenge

Authenticate end-user

- Collect username + password (or one-time token)
- Perform authentication check
  ➔ DO NOT STORE CLEARTEXT PASSWORD
  ➔ HASHED + SALTED = JUST GOOD ENOUGH

Keep user logged in

- Until she closes the browser window
- Until inactivity/absolute timeout

Solutions

Session-based authentication

- Do we have user authentication data in session data?
  - Yes – we're good
  - No – redirect to login form
- Is user authentication data valid?
  Check first and last access timestamp
  - Yes – update last access timestamp
  - No – redirect to login form

HTTP-based authentication

HTTP Authentication/Authorization Basics

Authentication might be performed by

- Web server (driven by configuration files)
- User script (parsing HTTP headers)

Typical authentication steps:

1. Client sends HTTP request
2. Server rejects HTTP request with error code 401 and WWW-Authenticate header specifying
   ➔ authentication method
   ➔ authentication realm
3. Client gets authentication data from the user
4. Client retries the same request with Authorization header

HTTP Authentication Example: First Request

```
GET /bin/start HTTP/1.1
Host: cmsdev.ipspace.net
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive

HTTP/1.1 401 Authorization Required
Date: Sun, 26 Feb 2012 17:28:28 GMT
Server: Apache/2.2.15 (CentOS)
WWW-Authenticate: Basic realm="development area"
Content-Length: 401
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

HTTP request without Authorization header is rejected

HTTP Authentication Example: Repeated Request

```
GET /bin/start HTTP/1.1
Host: cmsdev.ipspace.net
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Authorization: Basic VGhpczp3b250IHdvcmsK

HTTP/1.1 200 OK
Date: Sun, 26 Feb 2012 17:28:29 GMT
Server: Apache/2.2.15 (CentOS)
Expires: Sun, 26 Feb 2012 17:29:00 GMT
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

RFC2617: Basic = username:password encoded in Base64

HTTP Authentication methods (RFC 2617)

Authentication methods

- **Basic:** username and password encoded in BASE64 (totally insecure)
- **Digest:** MD5 hashes are exchanged in HTTP requests/replies
  ➔ As secure as MD5
  ➔ Not supported on all browsers and servers
- Integrated Windows Authentication (aka NTLM)
- Client certificates in HTTPS

Usage guidelines

- Use **Basic** authentication only over SSL/TLS (HTTPS)
- Client certificate authentication is a mess

Sample Script-Based Authentication

Perl script (snippet)

```
my $q = CGI->new;
my $auth = $q->http('Authorization');
if (!$auth) {
  print $q->header(-type => 'text/html',
       -status => '401 No username',
       '-WWW-Authenticate' => 'Basic realm="My Site"');
  exit;
}
```

Apache server configuration:

```
RewriteEngine on
RewriteCond %{HTTP:Authorization} ^(.*)
RewriteRule .* - [e=HTTP_AUTHORIZATION:%1]
```

Web Page Moves / Web Site Name Changes

Typical Challenges

I have moved my web site from www.foo.com to www.bar.info

- I want old links (and bookmarks) to work
- I want to retain my Google rank
- I don’t want to have duplicate content (which hurts Google rank)

I have moved my files from example.com/foo to example.com/bar

- I don’t want to see broken links
- I may want to move back from /bar to /foo

I want to redirect visitors to country-specific web sites (ex: Google)

Solution: Redirect Status

Status codes

- 301 – moved permanently
  ➔ Cache the new URL
  ➔ Don't try the old URL on the next request
- 302 – found
  ➔ Temporary redirection
  ➔ Don't cache the new URL, retry the old URL the next time

Mandatory: **Location** header (new URL)

Recommended: **Cache-control: private** header

Optional: HTML content

Commonly specified in web server configuration

HTTP Redirection Example

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:9.0.1) Gecko/20100101
Firefox/9.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive

HTTP/1.1 302 Found
Location: http://www.google.si/
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Date: Sun, 26 Feb 2012 16:36:06 GMT
Server: gws
Content-Length: 218
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
```

Redirect To Another Domain (Apache Config)

```
<VirtualHost *:80>
  ServerName  www.ioshints.info
  ServerAlias ioshints.info
  ServerAlias wiki.ioshints.info
  ServerAlias ipspace.net
  RedirectMatch 301 ^/$ http://www.ipspace.net/Main_Page
  RedirectMatch 301 (.*) http://www.ipspace.net$1
</VirtualHost>
```

Redirect To Another URL (Apache Config)

```
<VirtualHost *:80>
  ServerName  cms.ipspace.net

  RewriteEngine on
  RewriteRule  ^/$     /bin/start [R]
  ...
</VirtualHost>
```

Redirect to Login Form

Challenge

- Use session-based user authentication
- Redirect user to login form if
  (A) user just arrived at the web site
  (B) we lost session data
  (C) we detect authentication timeout

Redirect to Login Form

```
<?php
if (!$_SESSION['username']) {
  header("HTTP/1.1 302 Need to log in first");
  header("Cache-Control: no-cache");
  header("Location: http://www.example.com/login.php");
  exit;
}
?>
```

Usage guidelines

- Add original script URL to **Location:** header (don't forget to use URL encoding)
- Use **Cache-Control: no-cache** to prevent browser-side caching of the redirect

Questions?