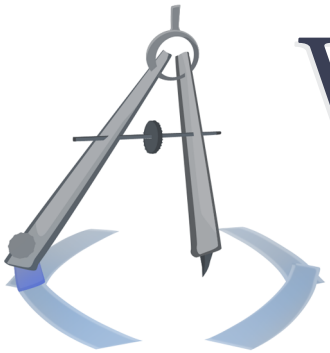


Website Architecture

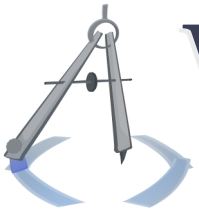
Project Specification

Project 3

The specification for Project 3, which tests proficiency with maintaining and modifying basic data-driven websites.

Michael Serritella

Summer 2012



Website Architecture

Project Specification | Project 3

Project Goals

This project tests data- and database-driven website design, including file-upload management, HTTP-header writing, and some basic measurement and tracking of user activity via digital forensics.

General Description

The project is a one-time-use file-upload service. The user uploads a file and receives a unique URI for the file. The user can give the URI to another user, who can download the file only once. After that download has completed, the file may not be downloaded again from that URI. The uploading user may see statistics on a private page regarding the time of download and the digital forensics of the downloading user.

In this way, it is similar to a "dead drop" in espionage parlance, and somewhat similar to a "burn bag" in that the sensitive contents are disposed of; the name "burn bag" is less morbid-sounding and more cool, so that is a nickname for the project.

The project shall use PHP to handle file uploads and a SQLite database to manage persistent state.

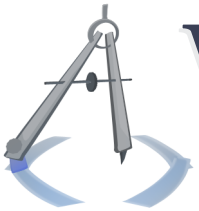
Provided Materials

All provided materials are on the course website in a file called "CIS4930_WAM_2012Su_Project3_Provided.zip". This includes:

- PHP files, called "Project3_BurnBag.inc" and "Project3_BurnBag.php".
- A SQL file, called "Project3_BurnBag_schema.sql".
- The CSS file, in the folder "includes", called "Project3_BurnBag.css"
- The JavaScript file, in the folder "includes", called "Project3_BurnBag.js"
- Its sundry images, in the folder "images"

NOTE: In order to modify the SQLite database, the Web server's Unix account will have to have write permissions on both the SQLite file and its directory. You can easily accomplish this by setting the permissions such that any user can write to the file. From the Unix prompt, go to the directory of your SQLite database and type this:

chmod ugo+w <TheDatabaseFilename>
chmod ugo+w .



Website Architecture

Project Specification | Project 3

Requirements

This section details the components of your project along with their requirements, followed by general requirements which apply to the whole project.

Database schema

A SQL file has been provided which partially describes the schema; it is not complete as given. A SQLite v3.0+ database must be built for this project, and it must have the given schema as a subset of its final schema (i.e. columns, tables, and other structures may be added).

Homepage interface

The homepage of the site should be plain and innocuous, with the only advertised functionality being the ability to upload a file. That may happen on a separate page.

Uploading a file

There shall be a page which simply allows the user to upload a file. The upload may be of any file type but must be less than 5,000,000 bytes in size. The user may also give a filename which would be presented as the suggested "Save As" filename when the file is downloaded. The user may also check a box that reads "Use original filename", which is checked by default; if this is checked, the original filename of the file shall be suggested when downloading, and this checkbox overrides any explicitly specified filename. If neither is given, a random, program-generated string shall be used as the suggested filename.

Upon a successful form submission, the user shall see a page with a success message and two links: one which may be used to download the file and one which may be used to view statistics on the download. On an unsuccessful submission, the user shall see a page with a failure message and shall be allowed to try again; no pre-filling of values or other helpful GUI hints are required.

Downloading a file

When a user visits a URI given as a download link and the file has not yet been downloaded, the user shall see a simple page asking if he/she would like to download the file. If the user chooses to download the file, he/she shall be taken to another URI and the download shall begin with the

uploader-suggested filename. More specifically, a download dialog shall appear; maybe the user chooses not to initiate the download. As long as this dialog appears once, that may constitute a download and the file would become inaccessible after that point from the download URI.

When a user visits a URI given as a download link and the file has already been downloaded, the user shall see a simple page saying that no file has been found.

Viewing download statistics

The uploading user can view a page which has data on whether the file has been downloaded. If the file has been downloaded, it shall show a timestamp (with year, month, day, hours, minutes, seconds, and time zone; any time zone is OK but specify which), the downloading user's IP address, and the downloading user's user-agent string.

File storage and implementation

The uploaded files shall not be stored in a way such that they may be requested via HTTP and downloaded outside of this system; i.e. the user shall not be able to download the file in a way that is not tracked and compliant to the policies described elsewhere in this Requirements section.

The database file shall not be stored in a way in which it may be requested via HTTP; the SQL file also must not be stored in a way in which it may be requested via HTTP.

General requirements

These requirements apply to the project overall.

Use of libraries

You may use MiniArc but no other PHP library, since it is hard (or unreasonably annoying) to judge how each library may have contributed to the project requirements; at least MiniArc is a known quantity. You are not required to use MiniArc in your code, however.

In Apache, you may use any module. Be aware that they might not be installed in your environment by default. For grading purposes, you can assume that the installation or availability of a library will not be a problem. You may want to search for how to install a module, however; it's not that big a deal.

Value-added content

For fun or education, any benign content may be added to that which is required - except that which is required to not exist - and it would not incur a penalty.

Use of the HTPPS and surrounding architecture

This project uses the HTPPS in a way very similar to Project 2. Similarly, you could use it or sidestep it.

Peruse the documentation of the HTPPS and MiniArc to see if it can help you with file uploads.

Getting Started

This project is the smallest and the most straightforward. However, it probably wouldn't benefit you just to sit down and start coding (you might as well learn that lesson now). Here are some questions you should probably answer before coding:

1. Where are the files going to go?
2. What are the download & stats URIs going to look like?
3. What are the database-schema implications of these answers?

You may want to use the MiniArc facilities for capturing file uploads, or you may not. Since this project is practically the most simple case of file uploading, the library would be of marginal benefit. In any case, you should still read the PHP documentation on POST file uploading and know how it works.

Submission

This project has no preliminary due date; all students are eligible for infinite resubmission of this project. This project only has the final due date which all projects share.

You may submit your work in one of three ways; choose the first option which is available at the time you wish to submit:

1. A form on the CIS4930.com course website
2. A Blackboard site for the course
3. Email; **mas04m@my.fsu.edu**

The project must be contained within a folder called "Project3_*[Surname]*", where *[Surname]* is your surname, using only alphabetic characters, without spaces, with the first letter of each word in your surname capitalized within the representation. That folder must be contained within a .zip file, such that the folder is the top-level element in the zip file. The zip file must be named "Project3_*[Surname]*.zip".

In Linux, you can achieve this by navigating to the parent of your project directory and executing the command: **zip -r Project3_*[Surname]*.zip Project3_*[Surname]***

Future Work

This project has some decent potential for expansion. Here are some ideas for your future work which are not required for this submission.

- Limit to one download per user
- Password protection
- Email the URIs to the uploading user
- Only count the file as downloaded when the download completes (or at least figure out how this would work).