Design and Implementation of an extensible, modular, web-based class-room supplement 2008-2009

Andrew Hamilton
Thomas Jefferson HSST

June 9, 2009

Abstract

The web today is fast becoming what the phone book used to be, except on a global scale. Most companies are expected to have a website, ranging from a few words on the homepage to massive and dynamic websites like that of Intel (intel.com). Many of these, especially those run by smaller companies or individuals, suffer from numerous security flaws due to the complexity of designing a functional and fashionable website today. Blackboard, FCPS?s Course Management System, is an example of an application that may have become somewhat out of date with regard to security standards. The goal of this project is to develop a basic web design framework that will handle the basic security and design concerns while being easily extensible with modules in order to be customized for almost any purpose.

Keywords: software design, software engineering, application, web-based

1 Introduction

The Internet is the ?rst place many people go for information. Many Companies need to make as much information available online as possible. However, websites often suffer from a lack of common-sense security measures as well as basic accessibility standards. The goal of this project is to provide a framework with which will help the end user to easily build their site through the use of simple add-in modules while transparently handling the security concerns. The ideal web framework would be easy and intuitive to use, extensible, and secure, the last becoming more important every day. This project describes the design of an alternative, web-based, extensible web framework running on a standard LAMP (Linux, Apache, MySQL, PHP) server.

There are many choices that can be made in web applications including language (PHP, ASP, Ruby, Python) and data storage system (LDAP, Oracle, MySQL, Postgre SQL). All have their advantages and disadvantages. PHP is the current standard for web pages although Ruby and Python are growing. ASP is a proprietary Microsoft language which makes it less than ideal for web site development. The current database standard is MySQL, though the others are popular for various purposes. PHP and MySQL are available for both Windows and UNIX operating systems which means that applications developed using them are relatively easy to transport between environments.

2 Background

Many websites today are designed to ?look cool? incorporating lots of Javascript and flash in order to try and ?stand out? in the crowded Internet Unfortunately, these sites often have security and accessibility problems that are in many cases left unfixed in the name of flashiness. For instances, Intel?s site looks incredibly cool with the animated menus on top of a background image, however, the colors can sometimes be hard to read and separating the image from the menus can be difficult for some people. Also, traversing the site with a screen reader takes much longer because the textual representation of the flash is incredibly lengthy. Another example, Sun Microsystems, implements a login system wherein your user-name and password (entered on one page) are passed to the next page through the URL such that your password can be seen by anyone looking at the computer and is saved in your browser?s history in clear text. This enables anyone who knows you have a sun account to check your history and easily determine the password. This is a bug that is easy to fix but which someone has thus far apparently not considered.

3 TJ cubesat project

TJ's cubesat project is an ambitious initiative to build and launch a pico-satellite built by highschool students. Through the systems engineering elective, students work with Teacher Adam Kemp as well as mentors and advisors at Orbital Science Corp on building the satellite. The satellite is currently projected for launch in winter of 2009-2010. As part of its payload, it will carry a digitalker, a chip which will allow it to convert text strings to speech which will then be broadcast on amateur radio frequencies. The plan is to setup a website which will allow people around the world to submit potential strings for consideration. These can be uploaded to the satellite after approval by a moderator. As part of this project, I plan to build the necessary modules to allow my core system to serve as the backend for this system.

4 Preliminary Testing and Analysis

Currently I have Apache2 with mod_ssl and mod_rewrite, PHP5, and MySQL5 running on a Debian Linux server in the lab. This is all that my project requires to run. The currently working version of my code presents a login page and denies access to the site without a valid login. Once a user logs in, their IP and browser agent are recorded for security, and they are given a PHP session which is used throughout the site. Before allowing them to access each page, the kernel module checks that their IP and browser agent are the same (to protect against session hijacking) and that they have not been idle for a long time (to protect users who forget to logout). All of these features work with multiple accounts and the main page currently uses information from the database to display the user's name and "rank" (teacher, student, or administrator).

5 2nd Quarter Improvements

Second quarter, the focus of my project changed from a scrict blackboard replacement towards a more general and versatile web development framework. The focused goal of this framework is to run the TJ Cubesat website which will take text submissions, hold them for review, and then upload them via radio link to an orbitting satellite. As a result of the radio link, I have had to use a python wrapper script to interface with the radio because

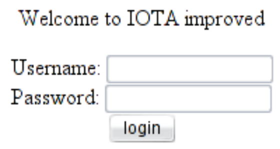

Figure 1: The login screen for IOTA, this is generated through the kernel module using the login module

PHP (a web-oriented language) has no built-in serial functionality. Building this script and making it work properly using PHP's exec() command was the primary difficulty I have had with making this alternative functionality work.

6 3rd Quarter Improvements

During third quarter, I have developed a python serial output program that allows me to interface PHP (which as a web scripting language, has no built-in serial functions) to the TJ Cubesat radio which has a serial interface. The current problem I am working on solving is the great OS divide since the radio is currently connected to a Windows XP computer while my PHP code is executed by a Linux Webserver. At this moment, it appears that the connection is going to have to be initiated by the Windows machine (there is no easy way to go the other way). While this is certainly the easiest solution, it does not allow me to implement live command sequencing to the satellite (while it is overhead). A second option is to separate the radio (which can be used on either linux or windows) from the antenna tracker (which requires a windows-only program) and put the radio on a linux box where two-way communication should be easier. A third option, which is a sort of compromise, is to have the windows machine request updates from the PHP script relatively frequently (say once every 10 seconds) and then act on them, perhaps with disregard for whether or not the satellite is overhead. This would allow for almost live command sequencing without the need for two computers on the remote enis to have the windows machine request updates from the PHP script relatively frequently (say once every 10 seconds) and then act on them, perhaps with disregard for whether or not the satellite is overhead. This would allow for almost live command sequencing without the need for two computers on the remote end. The disadvantage is that this will create a relatively constant stream of traffic and potentially radio broadcasting which could be potentially troublesome.

7 4th Quarter Improvements

In fourth quarter, I have finished packaging my security module for integration and testing with various sites. I also integrated it with a sample TJ cubesat website in preparation for TJStar day, although it was unable to run due to problems with the satellite hardware. I investigated various additional methods of providing site security, particularly for public parts of a website which cannot be restricted to a trusted group of users.

One of the problems with putting an email on a public website is that it is easily harvested by scavenger bots which download and scan the HTML looking for addresses to spam. An effective countermeasure that I have found is to run emails through an obfuscator, which replaces the regular ASCII characters with octal or hexidecimally encoded ones (such as amp; instead of an ampersand). Web browsers must understand this sort of encoding scheme and therefore are able to display a normal-looking email link. A bot searching the code, however, will see nothing that looks like an email address and therefore will be unable to harvest the email.

Another problem with public web-forms is getting the form spammed by some random bot that makes hundreds or thousands of bogus submissions in an attempt to overload a site. The solution to this is usually some sort of CAPTCHA (completely automated public turing test to tell computers and humans apart). These are those distorted images with text that you have to solve to do such things as register for a gmail account or post to some forums. The best of these systems that I have found is reCAPTCHA which was developed by Carnegie Mellon University. It soures words that could not be recognized by OCR from efforts to digitize old books and presents these as a CAPTCHA to the user. It both protects the site and helps to digitize old books which makes it a productive CAPTCHA. In addition, it is extremely easy for the end-user to implement; libraries are available for most major web programming languages. It also provides an audio system for those who cannot solve the visual one.

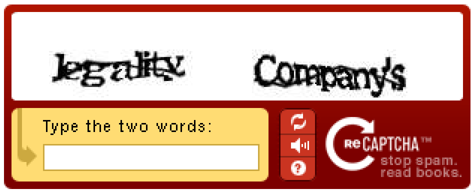

Figure 2: The reCAPTCHA input system

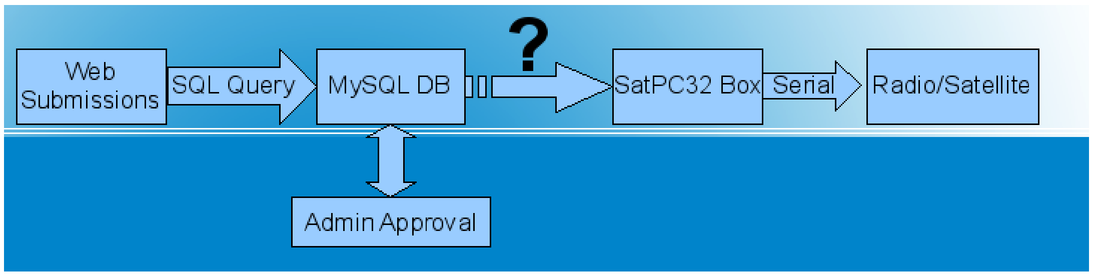

Figure 3: A diagram of the planned communications process

8 Cubesat Communication

The overall purpose of this program, at this point, is to send commands to the TJ Cubesat Satellite and receive data back from the satellite and make at least part of it publicly available. The current plan for the cubesat calls for commands to be short sequences of two or three letters or numbers. A colon will separate the command from it's arguments, the arguments are comma-separated, and a semicolon ends the whole set. The website will also need to take public input of quote suggestions for broadcasting using the Texspeak Text-to-Speech module on-board the satellite. These inputs need to be held for moderation until they are approved and it is possible that some sort of time-slot request system will be added (in which case the quotes would also need to be ordered). All of this information will be held in a MySQL database which will be accessed by the script and possibly directly by the python radio script. Telemetry Data downloaded by the satellite will also be feed into the database and likely made publicly available on the website.