

THEORY HITS THE INFORMATION HIGHWAY

A THESIS

Presented to the University Honors Program

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the

University Honors Program Certificate

Paul M. Nguyen

Spring 2010

**WE, THE UNDERSIGNED MEMBERS OF THE COMMITTEE
HAVE APPROVED THIS THESIS**

THEORY HITS THE INFORMATION HIGHWAY

BY

Paul M. Nguyen

**Alvaro Monge, Ph.D. (Thesis Advisor)
Computer Engineering and Computer Science**

**Burkhard Englert, Ph.D. (Thesis Advisor)
Computer Engineering and Computer Science**

**Timothy Caron, Ph.D. (Director)
University Honors Program**

California State University, Long Beach

Spring 2010

Overview

1. Acknowledgments.....	iv
2. Foreward.....	v
3. CECS-491 Final Report: Moodle+DB2.....	1-26
4. CECS-478 Final Report: Encryption in the US Government.....	1-16
5. Conclusion.....	vi

Acknowledgments

I would like to thank personally my wonderful mentors, Dr. Alvaro Monge and Dr. Burkard Englert in Computer Engineering and Computer Science for coaching me through the entire process, and Dr. Timothy Caron, Director of the University Honors Program, with whom I have had the great pleasure of working these past few years.

My interest in Open Source technology and in Computer Security began and flourished with my study abroad experience in Switzerland in the Summer of 2008, a partnership between California State University, Long Beach, Arizona State University, San Jose State University, and the Swiss Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (University of Engineering and Management of the State of Vaud). Through this program, for which Dr. Monge was instrumental in planning and executing, I was exposed to formal instruction concerning open source development tools and practices, as well as the intricacies of studying computer security, particularly cryptography. It was these experiences that sparked my fascination with this technology and led me to apply to doctoral programs around the country with the emphasis in computer and information security, and to pursue further studies in these areas for my culminating undergraduate work, represented by this thesis.

I would also like to personally thank my closest friends who encouraged me along the way, my teammates and partners in CECS-491 and 478, and my family for always being so supportive.

Katie E. Johns
Marcell D. Cadney
David A. Stout
Nathanael Trimmer

Marcus Graham
Tony Love
Maria Santella

Chan & Kathryn Nguyen
Christina Nguyen
Teresa Nguyen
John Nguyen

As I submit this thesis, my life is taking on a new direction and purpose. I have decided to pursue formation for the Priesthood in the Roman Catholic Church. This has always been a nagging possibility, since early in my Secondary School years, and it is something that I have always felt strongly about, though captivated in the moment by the very real obligations of my education at hand. Based on a final visit to the seminary of the Oblates of the Virgin Mary (OMV) in Boston, MA, I resolved to enter this coming Fall (of 2010). It is not without sadness that I proceed, but knowing that everyone who has been a part of my life up to this point cares deeply for my success and happiness and knowing that I have contributed positively to theirs places me in the warmest of spirits moving forward.

Thank you.

Foreward

This thesis truly brings the totality of skills gained in my studies in Computer Science to the table in a great synthesis that is this culminating work.

In the Computer Engineering and Computer Science department, dynamic web programming in PHP is taught only for the Web Technology Literacy Certificate, and not in mainstream classes. The Moodle project, however, required that we program in PHP along with XHTML and CSS, combined with knowledge of the DB2 Relational Database Management System. The goal of the project was to adapt the existing Moodle platform (a web-based learning management system) to use IBM's DB2 database system, in addition to the currently-supported databases. All of these technologies are fairly ancillary to the core curriculum of the program, however, the ability to move between languages and systems is a core skill that graduates from a Computer Science program must possess. Executing this project in a team environment with accountability to Dr. Monge and the developers at IBM required full application of our training in software engineering and technical writing. In this respect, participation in this senior experience is absolutely necessary to succeed beyond the walls of this University.

As far as my inquiry into Computer Security is concerned, it is primarily an application of all that we are taught in political science, communications, and critical thinking applied to modern technological solutions that can provide for security of information and computing resources. In order to investigate the use of encryption in the US Government, it was necessary first to understand the function of the technologies available, the process by which they are understood and certified for Government use, and finally the methods by which they are prescribed and mandated for such use. Without any piece of this marvelous puzzle, it is undoubtedly clear that very little of such an investigation could have been possible.

CECS-491 Final Report

Moodle + DB2

Marcus Graham

Tony Love

Paul M. Nguyen

December 16, 2009

CECS-491, Dr. Alvaro Monge

California State University, Long Beach

This page intentionally left blank.

Table of Contents

Introduction.....	5
Nature of the Project.....	5
Moodle.....	5
Motivation for Choice of Project.....	5
Open Source Software Development Experience.....	6
Internal Communication.....	6
External Communication.....	6
Environment & Tools.....	6
Server.....	6
Integrated Development Environment (IDE).....	7
PHP Debugger.....	7
Source Code Management.....	8
Bug tracking and development workflow / team dynamics.....	9
Interaction with proprietary stakeholder.....	9
Project Planning and Execution.....	10
Accomplishments.....	11
ADODB Testing.....	11
XMLDB Generator Class.....	11
Moodle modified to install with DB2.....	12
Unit Test Issues.....	13
Incompatible SQL.....	13
CLOB to text string comparison.....	13
Public Instances of Moodle Created.....	13
Other considerations.....	13
Open Source Contributions.....	14
Moodle.....	14
ADODB.....	14
Shortfalls & Transition Report.....	15
Moodle Core Modification.....	15
ADO-DB source file modifications and testing level.....	15
Moodle modification and patch status.....	15
Advice for continuing the project.....	16
CLOBS.....	16
Operational testing and remaining bugs.....	17
Personal Reflections.....	18
Marcus Graham.....	18
Tony Love.....	19
Paul M. Nguyen.....	20
Appendix A. Scope Document for Moodle DB2 Conversion Project.....	21
Table of Contents.....	21
Revision History.....	21
1. Project Requirements.....	21
1.1. Background.....	21
1.2. Project Goal.....	22

1.3. Project Objectives and Success Criteria.....	22
2. Scope and Limitations	22
2.1. Scope.....	22
2.2. Limitations and Exclusions	23
2.3. Project Implementation Plan.....	23
Plan A - Make it work, then fix the installer. (Preferred plan.).....	23
Plan B - Debug the installer, then verify functionality.....	24
2.4. Project Change Request Handling.....	25
Appendix B. Project Links.....	26
Core Project Links.....	26
Project Documentation.....	26
Reference Material.....	26

Introduction

Nature of the Project

This class project is designed to give students the chance to participate in real-world software development, via a contribution to an open-source project. Teams may choose from a few well-established open source projects that are proposed to the class; they may also opt to select another project, subject to approval by the Professor. The class project is to be characterized and proposed by each team, and should involve either an enhancement or bug fix, desired by the community, that is sufficiently complex to occupy the students' time for the duration of the semester, and reasonably manageable to assure completion by the end of the semester. Students are requested to document their intended work, their progress, and their accomplishments throughout the project, and present their work to the class at the close of the term. Students are required to participate fully in the open source community pertaining to the project of their choice, including communication with the community throughout the term, and a submission of the code produced by the end of the term.

Moodle

Moodle is an open source Course Management System (like CSULB's BeachBoard). Moodle can be used by anyone, from primary schools and educational hobbyists to large universities with hundreds of thousands of students to school district administration with multiple schools. It can be used for fully online classes or as a tool to augment face-to-face learning – in what is called "blended learning." Moodle provides many communication and collaborative tools such as forums, wikis and databases. It can also be used to deliver content to students and even administer assignments and quizzes. These features and more can be taken advantage of, as the teacher wishes, to create a completely immersive learning environment, in and out of the classroom.

Motivation for Choice of Project

There were multiple reasons for choosing this project. First, Moodle is a mature project. According to ohloh.net, it has 732,726 lines of code (50 Mb of PHP) and an \$11 million lifetime cost. This project also provided us with an opportunity to work in conjunction with IBM. We saw this as in advantage because it gave us the opportunity to work in an open source and corporate environment. Also, this allows us not only to mention contribution to an open source project on our resumes, but also involvement with a well known company in the computer industry (IBM). Finally, we were drawn to this project because switching from BeachBoard to Moodle has been an on-going discussion at CSULB, so we saw the project as a potentially relevant contribution to our own school community.

Open Source Software Development Experience

Internal Communication

The group's initial form of communication was to trade email addresses and phone numbers. We also used different software products for multiple person communication/conference calling. One such software was DimDim.com, which is a web browser-based communication software that allows audio communication, instant messaging, and even has a shared white board. Another software used was Skype, which allows audio/visual communication and instant messaging between multiple users.

The main platform for communication between the group members was using Google products. The IBM project lead created a Google Group for the project, which was handed over to the group for communication and file-sharing purposes. The Google Group Discussion section created a valuable resource, not only for communication but for documentation and future problem-solving as well. The Files section was used to post static files for each of the group members, and the Pages section allowed for the group members to each document their personal contributions as well as a page for project links.

Google Documents was also used by the group to share documents between group members, and make documents publicly available to other members of the project, such as Dr. Monge (the instructor) and the team members at IBM. Some of the files stored in Google Docs are the two presentations (project selection and scope), the scope, the timeline for the project, the status reports, and the draft version of this final report. The editing rights to most of these documents are shared by all of the group members, and many of them are publicly viewable (with links to them posted in the Google Group).

External Communication

For communications between the team and other groups, we employed the standard communication media for each party. We used Moodle and ADODB forums, documentation wiki user pages, and mailing lists to announce, track, and inquire about various aspects of the project. These methods, being the standard for such communities of developers, proved to be very efficient, with multiple credible responses from project leads received within the first day or two of a posting. The exception to the response time, however, was ADODB. There was very minimal response to posts from our team to their forums. This reality, though frustrating, is understandable, given the volunteer nature of open source; a project's owner is quite likely to be on vacation or attending to family business, or whatever else may take him away from the project, delaying his response to communications from the public.

Environment & Tools

Server

After much deliberation and experimentation with virtual machines and a preliminary look at Amazon Web Services, we opted to proceed developing Moodle on our personal computers. To

do this, we used the free web server bundle: XAMPP. This package includes an easy-to-use installer and promptly provides Apache, MySQL, PHP, and Perl to the Windows desktop. With a little configuration, Moodle was installed and running on the included MySQL service. With further PHP configuration and the installation of IBM DB2 Express-c, we began to attempt and later verify installations of Moodle on this platform.

As work on the project continued, it became necessary that we use a publicly-accessible test-bed. The answer was Amazon Web Services (AWS). Through the IBM Academic Initiative, we were able to obtain credits that covered our usage of AWS, and provided us with specific tools that were required of a web server hosting a PHP application, driven by DB2. This consisted, primarily, of a Linux operating system, the Apache web server, PHP, the MySQL server (again, as a reference installation), and DB2 Express-c version 9.7. Through our partnership with IBM, our public test instances of Moodle were connected to the domain learndb2.org, as mentioned elsewhere in this document. This accessibility will hopefully encourage larger-scale debugging and issue-identification that would not otherwise be possible while working in a small team environment.

The use of open source tools on servers can be daunting, and even nerve-wracking to consider, because the system administrator, though he does have access to the source code of the software products he will be using, cannot possibly review all of that code with sufficient scrutiny to identify defects or flaws that lead to vulnerabilities. Instead, the system administrator trusts in the greater community to identify, isolate, and fix such imperfections with sufficient speed to outpace would-be attackers. The other daunting task facing system administrators who deploy open source software is the task of installation and configuration of the server software. In many cases, including ours, several server software products had to communicate with one another to support our application. The Apache web server had to process the PHP scripting, which called special functions in built-in libraries that accessed databases and processed files present on the server. Each software layer worked with the others on every request to execute everything specified in the application.

Integrated Development Environment (IDE)

The choice of a functional, intuitive source code editing application was one of the more important environment decisions the team had to make. We initially considered using Eclipse for PHP; this consideration was time-consuming and ultimately, we followed instructions on the Moodle development site, indicating that NetBeans is a more appropriate product to use for this purpose. NetBeans integrated an FTP client and Source Code Management tools, in addition to providing a very easy-to-use code navigation system. All of these features were indispensable in enabling faster work toward project goals.

PHP Debugger

The PHP debugging process was slow and somewhat blind. PHP is an interpreted scripting language that is triggered by an HTTP request for a page that the web server identifies as requiring pre-processing by the PHP engine. Because the execution is remote, it is difficult to track the progress of execution step-by-step, as in the traditional compiled or locally-interpreted language systems. Rather, special software must be configured alongside the server, and a special

connection must be established between the debugger and that server-side module, in order to debug (in the traditional sense) a PHP script.

For Moodle, this was a tremendous aid, because each request caused a dozen (if not more) PHP files to be strung together, making it very difficult, by the rudimentary method, to determine at which point a failure occurred. The Zend debugger, paired with the Eclipse IDE, proved to be the right tool for this job, and, once employed, helped resolve the `update_record` issue (detailed later) that brought progress to a halt for a significant period of time.

Source Code Management

Source Code Management (SCM) is a paradigm upon which Open Source software development depends fully. Open source requires that many coders world-wide be able to contribute to a project, both by viewing and submitting code, and a system is necessary to provide this functionality.

The first generation of SCM was a client-server model with locks. This means that there is an authoritative source for the history of a project's code's development (the server) and that when programmers (clients) wish to change the code, they had to obtain exclusive rights to modify that code. The system would guarantee that two people could not post their changes at the same time, or without consulting each other, and that would maintain a lossless and complete history of the development of the project.

As projects grew, this method became impractical, and the first very popular SCM hit the market: the Concurrent Versioning System (CVS). CVS solved the problem of exclusive write access by requiring that clients synchronize their changes with the latest changes available on the server before submitting them; this allowed programmers to work at their own pace and for programmers located at great distances from one another to contribute to the same project with little need for direct coordination. Moodle uses CVS as its SCM, and to obtain the Moodle code, we used both development snapshots, as well as CVS checkouts. CVS was superseded by Subversion (SVN), which provides better fault-tolerance on the server-side and more sophisticated and modern methods by which to access the code. Moodle has remained with CVS, however.

As is evidenced by the transition of several prominent open source projects, a new paradigm is taking a foothold for the long-term: distributed SCM's. Among the most popular of these are git (developed for the Linux kernel by Linus Torvalds), Mercurial (or Hg, written in Python), and Bazaar (or Bzr, also written in Python). The fundamental difference is that each client maintains a complete history of the project (clients using centralized SCM's only store relevant snapshots of the code history, and can make do with only a single "checkout" of code – the state of the project at a single point in time). Thus, in the distributed model, each client is a peer, and there is no server.

Because of distributed SCM's prominence in the marketplace today, our team elected to make use of Mercurial for this project. We used the free BitBucket Mercurial hosting service, which is web-based, and includes a syntax-highlighted view of our current source code, as well as a Wiki for documentation purposes and an Issue tracking system to help us share our pursuit of bugs, defects, and necessary enhancements to the project as we went along. The Issue tracker became

immensely useful for IBM developers to point out flaws in the public installations that we made available to them for testing.

Bug tracking and development workflow / team dynamics

The process of identifying coding tasks, bugs and imperfections, and other code-relevant points of inspection needed to occur concurrently. That is, for the team to be effective, we needed to be able to share information gathered (possibly simultaneously) in a central location from which we could see the outstanding issues, and assign them to ourselves to ensure their completion.

The solution was BitBucket's Issue Tracker. This tool allowed us, as administrators of the project, to submit issues and manipulate them, assigning ourselves to them, updating them with direct links to source code, classifying them into our timeline, and resolving them, as needed. The system also allows members of the public to submit issues; this feature became invaluable when we released the public installation of Moodle on AWS and IBM developers had the opportunity to file tickets in the system based on their experience with the live product.

Over the course of each week during the semester, the co-curricular and extra-curricular obligations of each team member never quite coincided, leaving our availability largely disjoint. For this reason, much of our development was never done in parallel; though we worked on separate tasks, we rarely worked at the same time, and almost never at the same time and on the same task. This is one stark contrast to the typical conception of open source software development, which is, at times, fast-paced and massively-concurrent.

It was also necessary for us to be able to identify trends in the issues filed, or other research-based discoveries made about the technologies involved. For example, learning about DB2-specific functionality with respect to reserved words led to the consolidation of several issues into a single point of failure that could then be solved, reducing the apparent cloud of problems significantly. The magic quotes PHP setting (described later) was another source of multiple defects in our system that, once we did our reading and isolated testing, attributed this incorrect setting to multiple frustrating and apparently-disparate failures. It was precisely this ability to lay everything on the table that enabled us to see the bigger picture and connect seemingly unrelated issues in order to solve them.

Interaction with proprietary stakeholder

As mentioned before, a Google Group was set up by the IBM project lead, and control of the group was handed over to the group. This group began as public, but was eventually changed to private due to spamming. Despite being private, membership was provided to all IBM employees involved in the project. This way, the IBM stakeholders had access to all discussions between the group members and could provide input or advice to any discussion between the group members. Also mentioned above is the group's status report, which was kept in Google Docs and updated weekly. When each status report was completed, the contents would be posted in the Google Group for the IBM members as well as the instructor to view.

Besides the weekly status updates, the group also kept in touch with the IBM team through weekly conference calls. These conference calls would take place on the IBM conference calling system, and would involve the group members, IBM group members, and the class instructor. The calls would focus on that week's status report, which consisted of the previous week's

accomplishments as well as the next week's goals. Other topics would be the project timeline and scope and their relationship to the current status of the project.

One of the catches to the Google Group was the IBM group members' inability to view the Moodle code. This meant that any discussion of actual code from the Moodle code base would have to take place through email, or some other medium outside of the Google Group. The IBM members were free to discuss concepts and give pointers as to possible solutions based on problems we discuss in the Google Group. The IBM members could not provide any proprietary code from the IBM end either, as it would compromise the open source code of Moodle. This meant that we could discuss the functionality of IBM proprietary software, such as the DB2-PHP driver, but could not view the code directly. Basically, any conceptual or functionality discussion was allowed, but the group was not able to view any IBM code, and the IBM members were not allowed to view any Moodle code.

Project Planning and Execution

After exploring the function and purpose of Moodle and DB2, as well as build familiarity with the tools and languages involved, we worked on our project scope and definition, included in Appendix A. At the time, we determined that there were two routes to follow.

Plan A realized a goal and overstepped a hurdle by taking an existing Moodle installation on MySQL and migrating it to DB2, that is, taking the table structure and data from MySQL and recreating it on DB2. After this was complete, we would change the Moodle configuration so that it would access the data stored in DB2 rather than that stored in MySQL. This plan proved to be a bit troublesome initially, and with only minor successes, the team decided to pursue completion of the project via Plan B.

Plan B takes a direct approach to the problem, solving each issue in sequence. It begins with modification of the installer and requisite other components, especially the XMLdb generator class. It then proceeds to specify how various testing tools will be used to determine further incompleteness in the compatibility with DB2.

Both plans were divided into phases that included time constraints, anticipating delay and reserving time for documentation efforts.

Accomplishments

ADODB Testing

A subset of the pages found in the ADODB/test directory were modified to provide initial testing of the DB2 native driver version 1.8x. This testing demonstrated that DB2 was indeed compatible with ADOdb in all areas except one, the serverinfo function. This function is found in /adodb/drivers/adodb-db2.inc.php starting at about line 198. This file is a modification of the original ODBC version of the ADOdb DB2 driver. The serverinfo function still refers to an ODBC data source and will need to be rewritten to support the DB2 native driver. Issues with Moodle and the serverinfo function will be addressed in a later section. Otherwise ADOdb proved to be fully functional for our requirements.

The test pages modified are detailed here:

File Name	Function
benchmark.php	Cache throughput test
test-active-record.php	Record set functions against a database.
test-datadict.php	Informational – table features
test-php5.php	ADO-DB php5 capabilities test
test-xmlschema.php	DB data to MS SQL.
test.php	Basic functions test
test4.php	INSERT/UPDATE/CREATE/DELETE test
test_rs_arrays	ADO-DB test
testdatabases.inc	many tests
testgenid	as named
testpaging	as named
testsessions	as named

These tests are suitable for verifying the basic functionality of DB2 and ADOdb.

XMLDB Generator Class

The Generator Class (lib/xmlldb/classes/generators/db2/db2.class.php) is used by Moodle to take the tables created in the XMLdb layer and generate them in the RDBMS backbone (DB2, MySQL, PostgreSQL, etc.). This file sets database specific requirements, such as default values, what types of keys and indexes are allowed, and SQL for specific functions. The two most important functionalities that we have perceived so far are the setting of data types and handling of reserved words. The data types are set in the function getTypeSQL. DB2 is extremely lenient on the use of reserved words, so our current implementation treats reserved words the same as any other table or field name. This was achieved by filling the

`$reserved_words` array with the list of reserved words from the DB2 manuals, but setting the `$quote_string` variable to an empty character. This keeps the generator from putting quotes around reserved words, a functionality which led to multiple `INSERT` failures during the install (since a lot of the installer's SQL is hard coded into Moodle).

The generator class inherits from the generic XMLDB Generator Class (`lib/xmlldb/classes/generators/XMLDBGenerator.class.php`) and overwrites what is specifically necessary for DB2. To help with reference, all overwritten variables and functions include online references, and the original XMLDB Generator Class was copied and pasted into the DB2 Generator Class and commented out. This "debugging" version of the generator class is still included in the main code base since the project has not been technically finished, and these comments can be removed upon completion of the DB2 adaptation.

Moodle modified to install with DB2

This required the modification of several files to accomplish.

First – `/lib/setuplib.php` was modified to test detect a proper UTF-8 database prior to installation. This entailed adding a case for DB2 and the specific SQL to check the value.

The file `/lib/environmentlib.php` was then modified to allow the installer to complete. A version check in this file at lines 807 through 822 fails. This failure is due to the `ServerInfo` function of `ADODB` not being implemented for the native driver. A modification to the `else` clause of the `if` statement (`false` to `true`) allows the installer to complete. Bug 7 posted in BitBucket documents this. Our recommendation is not to fix this.

Several instances of text comparison failure were found during install. This is due to DB2 returning column and data table names in upper-case format. Moodle compares everything to lower-case strings. These types of failures were an issue on two occasions that required lengthy analysis to isolate. After identification of the issue, a fix for column names was implemented that provided alternate upper case strings to use for DB2 comparison. This applies to `lib/dmllib.php` at lines 1685 to 1722. The setting `ADODB_ASSOC_CASE` controls the casing of table names and must be set to `0` for `ADODB` to function correctly with DB2.

Some other SQL specific issues led to failures during the installation. One such issue was the use of the `LIKE` predicate. DB2 does not allow anything but string constants (or a regular expression or concatenation of string constants) on the right hand side of the `LIKE` predicate. This was fixed in `/lib/accesslib.php` (starting line 1274 through 1312) by using the `INSTR` function (references sited inside of function).

Another fix in `/lib/accesslib.php` was to accommodate `UPDATE` and `TRUNCATE TABLE` calls. The `UPDATE` was fixed by including a DB2 specific `UPDATE` SQL call (`/lib/accesslib.php`: lines 5481-5509, fix starting line 5495), and the `TRUNCATE TABLE` (which is a function that does not exist in earlier versions of DB2) was replaced with a DB2 equivalent (at least for the purposes of our implementation) in `/lib/accesslib.php` (lines 5511-5517, references cited in code).

After the completion of these modifications, Moodle consistently installs correctly on DB2.

Unit Test Issues

Upon successfully completing the installer modifications the Moodle unit tests were evaluated. There were several unit tests that failed due to the following reasons:

Incompatible SQL

There were two SQL incompatibility issues. The first was the use of the SERIAL keyword in the Unit Tests. This is a mySQL keyword that sets up an auto-incrementing integer field (generally an id field). This was fixed by including a DB2-specific statement with equivalent functionality (similar to that found in the DB2 Generator Class). The other issue was the use of the backslash (\) as an escape character by most databases and PHP, but *not* by DB2. This fix is explained in more detail in the "Other considerations" section.

CLOB to text string comparison

This issue is evidenced by the Grade_Item unit test failures as well as multiple debug traces. This is due to a text string value being compared to a value stored in a CLOB field. This is unsupported in DB2. To verify this was the issue an experimental version of Moodle was installed with a modified generator class. All CLOB fields were redefined to varchar(2500) and the database table page size was increased to 32K. The size of 2500 was arrived at by experiment and represents the maximum size that will allow a successful install. This configuration clears up all CLOB comparison-related debug dumps as well as the Grade_Item failures. The experimental version of Moodle is located at www.learnadb2.org/expmoodle.

Public Instances of Moodle Created

To facilitate testing Moodle was installed in an IBM development AMI on Amazon Web Service. The base url is www.learnadb2.org. Three versions of Moodle are installed.

At /moodle is a version of Moodle representing all the fixes to date except for CLOB handling.

At /expmoodle is a version of Moodle with all our current changes in addition to having text fields redefined to varchar(2500) and a table page size of 32K. This version is intended for evaluation only and is not presented as a possible fix for the CLOB problem.

At /mymoodle is a version of Moodle installed on MySQL using the 1.9.6 code base.

Other considerations

A problem was found with Moodle during install a subsequent testing that had two symptoms. First was \ ' to ' ' conversion. The ' ' is the correct escape sequence for DB2. The second symptom was the addition of a \ every time a \ was input in a text field and submitted through a form. We corrected both of these with the php.ini setting magic_quotes_sybase=0n.

Open Source Contributions

Moodle

Based on advice from Moodle developers Tim Hunt, Martín Langhoff, and Petr Škoda, we pursued the route of making the necessary changes to Moodle, and then posting them as a contributed module.

Our patch, based on the 1.9.6 stable release of the Moodle core, is now posted in the Modules and Plugins Database on the Moodle site (<http://www.moodle.org/mod/data/view.php?id=13&rid=3100>), and contains links to our development resources, including BitBucket and the Google Group.

ADODB

Based on testing done with ADODB itself, and then the performance of ADODB as included in Moodle, we believe ADODB to be sufficiently complete for use with DB2, with no modifications. Posts have been made on the ADODB forum (<http://phplens.com/lens/lensforum/messages.php?id=18261>) that detail our findings and provide additional information to the maintainers of ADODB. Since the project began, however, we have received no feedback, either to our initial posts that sought to determine the requirements for achieving a support level increase, or to our final posting that we see fit for DB2's support in ADODB to be elevated.

Shortfalls & Transition Report

In general, the project has been successful in getting Moodle adapted to DB2. The demo site (www.learn2.org/moodle) appears to have attained functionality on par with a MySQL-based Moodle 1.9.6 reference implementation (www.learn2.org/mymoodle). However there still bugs outstanding concerning serious failures in core functionality, and the modified Moodle code is not well-tested. Other aspects of the scope were not met as well, such as submitting project changes in Moodle to be included in the current 1.9.x code stream.

Moodle Core Modification

Section 1.3, Item 2 of our scope specified that all code changes introduced would be submitted back to the Moodle community, to be included in subsequent 1.9.x releases. This proved impossible to accomplish, as the Moodle community would not accept it. The primary reason is the concern that users installing with DB2 would not have an upgrade path to the soon-to-be-released version 2 of Moodle. The Moodle developers then recommended that we instead create a patch and submit it to their plugin database. This is what the team decided to attempt.

Since we still have outstanding bugs, we have decided to submit an "install" version of Moodle to the patch database. This will include our code that does not address the CLOB failures that exist, so it is suitable only for testing.

ADO-DB source file modifications and testing level

Section 1.3, Item 3 of the scope requires all ADOdb source modifications be submitted to and accepted by the community. This was not necessary; the project did not require changes to ADOdb core files to achieve compatibility with Moodle and functionality with DB2. Instead some of the test pages included with ADOdb were modified to assist in the verification of the DB2 native driver with ADOdb. This testing found no defects in the core functionality, the exception being the ServerInfo function as detailed in ADOdb testing above.

Section 1.3, Item 4 requires that the testing level of ADOdb be elevated from level C to level B as defined by the project owner. The levels are defined as:

B = tested and usable, but some features might not be implemented

C = user contributed or experimental driver. Might not fully support all of the latest features of ADOdb.

In our testing we found that ADOdb is usable, and has a feature unimplemented. This seems justification to request the level change. Our findings concerning ADOdb have been posted in the ADOdb forum along with a request to have the testing level changed. To date, the project owner has not responded.

Moodle modification and patch status

The modification of Moodle to utilize DB2 is not complete. The outstanding issues will need to be addressed by the next team to follow. In the bug tracker, bugs 16 (CLOB comparison) and 20 (SQL0401N) are both related to the CLOB comparison failure; bug 19 (Clicking on

Reports/Security overview results in DB2 CLI errors) is a SQL syntax error; bug 7(/lib/environmentlib.php) is the version comparison error detailed earlier. There are many reasons the modification is not complete.

A team member withdrew from the project unexpectedly. This loss of man-power proved to be a critical factor in the efforts for Moodle.

There were several problems encountered in the installer that required extensive analysis to isolate. This set the project on a schedule of fixing one hard problem per week, which consumed substantial time. The velocity of the project slowed further as the semester progressed and the demands of our student schedules had to be met.

Lack of a reliable PHP debugger was a liability. Several attempts were made to utilize Netbeans and Eclipse as a PHP debugger. Only limited success was attained using Eclipse with the Zend debugger. This success was enough to isolate the very stubborn update_record failures that held up our efforts for two weeks.

Advice for continuing the project

CLOBS

The first issue that a continuing team needs to resolve is that of the CLOB comparison failures. This failure can be witnessed via the unit tests of the grade items tables. Bugs 16 and 20 are both due to CLOB comparisons failure.

Please see the MySQL and DB2 documentation for an explanation of text and CLOB fields.

Moodle is intended to be used with many types of database servers. To accomplish this, the Moodle developers have implemented a system known as XMLDB. This allows Moodle to be designed conceptually using XMLDB constructs as database objects. Access to the underlying database server is done in a generic way. This works very well, and the problem of adapting Moodle to a new database server becomes that of mapping the XMLDB object to the correct object in the target database system. This is accomplished by a class known as the generator class. Located in /lib/xmldb/classes/generators/db2/, the file db2.class.php provides this functionality for DB2. Lines 1153 through 1218 provide the XMLDB to DB2 mapping that defines all the data types used.

Moodle has the concept of a XMLDB text field. This text field is analogous to the text field as defined by MySQL. This MySQL text field is similar to the DB2 CLOB field. The CLOB is the mapping to be used when implementing the XMLDB text field for DB2. Both MySQL text and DB2 CLOBS are stored in a location other than the table and do not increase the table size significantly. The problem we are encountering rests in an incompatibility between DB2 and MySQL. The MySQL text field is directly comparable to a text string in SQL; a DB2 CLOB field is not. It is this incompatibility that is at the root of the failure.

An approach to fixing this is to redefine some of the XMLDB text fields in Moodle's tables to be a varchar of some size. This the approach taken with the experimental version of Moodle found at www.learn2.org/expmoodle. The DB2 implementation of the XMLDB text field was mapped to varchar(2500) and the table page size of the DB2 database was increased to 32K.

This experimental version does not exhibit the comparison failures seen in www.learn2.org/moodle.

This is not the recommended fix, however. Instead of globally setting all XMLDB text fields to varchar, only selected Moodle text fields should be changed to varchar of some size. Some of the CLOB fields are needed to support large amounts of text that may be entered into some text areas. If all CLOBs are varchar(2500), some of the fields may not be usable due inadequate storage allocation. Instead, if only some text fields were changed to varchar a larger size could be used as more room would be available with the 32K page size.

Grepping `$field->setAttributes(XMLDB_TYPE_TEXT` in `/moodle` will reveal the extent of text fields in Moodle. The text fields are of three sizes – small, medium, and large. It appears the small text fields are id or reference fields while the small designator seems to infer that only a small amount of data is expected. It may be that these small text fields can be redefined as `varchar(x)` while leaving the medium and large text fields as the appropriate size CLOB.

If this approach is taken, the suggested implementation location is in the DB2 generator class. Currently the DB2 generator implements text fields as the default CLOB size regardless of size is requested. The text definition should be enhanced to detect the size requested and give a varchar for small text fields and the correct size CLOB field for medium and large text fields.

An alternate approach that does not remap small text fields to varchar could be attempted by casting the appropriate CLOB field to varchar. This would require extensive analysis of Moodle to find all the locations where text fields are compared to text strings. Instead of applying the fix to a DB2-specific file, many other Moodle files will need to be modified and tested.

These are only suggested approaches to correct this issue. The next team will need to consult with the Moodle development community as to the appropriateness of these or any other solution to this issue.

The CLOB issue is the blocking function to completing this project. This team did not have time to resolve this issue due to the need to analyze the required changes and consult with the Moodle community.

Operational testing and remaining bugs

Due to time constraints and man-power shortages, extensive operational testing was not performed. After resolution of the CLOB issue, it is recommended that comprehensive testing be performed to ensure all areas of Moodle are working correctly. It is believed that the CLOB issue is the last major issue to be resolved, with only minor bug fixes to remain. Subsequent errors will likely be either SQL syntax errors hard coded into a page or text comparison errors of metadata due to differences in text case.

Personal Reflections

Marcus Graham

I found this project to be both challenging and interesting. Having never worked with DB2, PHP, or Moodle I found that the initial research into them invaluable. Initially fault isolation proved difficult due to the lack of a reliable PHP debugger, but this was addressed with the use of debugging support provided by Moodle. Eventually Eclipse paired with the Zend debugger proved useful for this project.

The tools I used on this project included XAMPP, XAMPP for Linux, Eclipse with web tools support, Notepad++, Astrogrep, Filezilla, Putty and the Amazon Web Services. With the exception of AWS, all the tools were open source or community edition software. Other tools included use of IBM's migration tool kit to migrate a MySQL version of Moodle to DB2 for testing purposes. DB2 Control Center was used to work with DB2 databases.

This project gave me ample opportunity to dust off my troubleshooting skills and work on a project that actually intended to accomplish something. Working with the team proved challenging in many respects. The most trouble came from the inability of team members schedules to coordinate which led to a serial pattern of individual work. This caused progress to be slower that it should have been and led to the incomplete modification we are leaving for the next team.

I found working with the open source community a little frustrating as I attempted to coordinate with ADOdb project owner. Despite repeated attempts at contact via email and forum posting I have yet to receive a response. However, in general I found working with open source tools and contributing to the Moodle project a rewarding experience. I have never hesitated to use open source tools to accomplish a task, and will look for an open source solution to a problem before choosing commercial products.

I particularly enjoyed working with the Amazon Web Services. I feel this experience alone was worth the cost of admission. I would likely never have paid to use the service so it was fortunate we were able to get the student credit for experimentation. Implementing Moodle on an IBM AMI proved to be a great introduction to using the system. This also gave me an opportunity to practice my Linux admin skills. The AWS self provisioning ability is a great, though for a student pricey, solution to high costs associated with an Internet start up.

I found working with IBM overall to be an educational experience. All IBMers that I dealt with were very friendly, professional and a pleasure to work with. I regret that we could not finish the project for them.

Overall, this was a good opportunity to gel the things I have learned in my years at study with the skills I had attained in my previous careers. I definitely feel ready to go out among the employed.

Tony Love

I chose to take this class for multiple reasons. This class provided for me a number of opportunities that I had not received before and wished to take advantage of. First, was the opportunity to work on an already existing project. For me, much of the importance of this project was the focus on manipulating and improving a mature code base, as opposed to most class projects where the teacher has the student create his or her own project from scratch. I felt this was important because this is a much more common scenario in the real world than building your own project from the ground up.

This experience also provided me with the opportunity to work on a project that has some sort of significant effect in the outside world. Although I have completed many projects and have done fairly well in my studies, I have had little to no real world experience in the Computer Science industry. This experience gave me the opportunity to work on a real-world project, and, in the end, make a contribution to that project that I can cite on my resume.

This experience also gave me the opportunity to work with one of the leading companies in the industry in IBM. I felt this experience was invaluable because it gave me a glimpse of project structure and expectations in a real working environment. It also gave me the opportunity to network, meet people in the industry working for one of the industry leaders, and provided me with a big name developer to include on my resume.

Finally, the project provided me with valuable experience by working in a somewhat strenuous group environment. Although I have worked on group projects before, generally the group is able to meet in person and discuss problems, strategies, and group objectives, and in many cases were able to work together simultaneously. This was not the case for this project. The various conflict schedules, combined with periodic group dynamic issues, really forced me to better develop my networking and group communication skills. It also forced me to work on my scheduling, documenting, and parallel development skills.

Overall, this project definitely provided me with a truly unique and invaluable experience. Not only was I able to achieve all of my goals coming in to the project, but I was able to develop skills that I had not been forced to develop before, and, in many cases, believed that I already had. Even though we were not able to completely meet our objectives for the project, I was able to meet many of my personal goals set forth at the beginning of this project, and, in light of that, would still consider this project a success.

Paul M. Nguyen

I have been an avid user of open source software products for several years, to date. I have always sought an open source software solution before a commercial one, and have spent ample free time exploring Linux, BSD, and various free applications, including web-based applications, particularly those written in PHP and that use databases to store their content. I recently began looking into contributing to open source projects more heavily, especially after my study abroad experience in Switzerland, where I studied open source software engineering tools (like version control systems, patching, and build environment toolsets) and licenses, among other specialized topics.

I worked in teams developing software previously, but our projects were always self-contained, no matter how many open source tools or models we used. In fact, previous projects used the same tools that we used for this project, and the only difference, then, was that we were not accountable to the community. For this project, interaction with the Moodle community was very interesting. The developers were not very open to our would-be contributions as an essential part of the project, but they were encouraging and helpful in providing information that advanced our progress. This was a reality we had read about and had to face directly – that the core developers, the visionaries of a project, do not accept every idea that comes their way, even if that idea is backed by other developers who are poised to complete the development themselves. It was the discretion of the core developers that we were up against, and so we had to proceed with our development with the ultimate submission being a patch that modifies the core Moodle code to make it work with DB2.

Being fairly familiar with workflows for using source code management tools (Mercurial and CVS, in our case), and working with interpreted PHP, it was very easy to work on this project. It was interesting, however, working with my teammates and introducing them to these tools. Our communication via Google Groups was not foreign to any of us, and the forums and mailing lists that we used are fairly standard practice, also not something we had experienced previously.

Personally, I learned about the dynamics of working on a team that is accountable to a larger community of developers, and the dynamics of communicating with the core developers of an open source project. It is important, in both cases, to maintain respect, and to afford the others the same convenience that I would reasonably expect from them. This means that communications should not be executed sporadically, or without preparation – background research and assembled knowledge, references, and other supporting information should be included, in order to prove that I have invested in the subject. It is also important not to be discouraged by initial nay-saying. It was the case, with this project, that we were denied contributory privileges to the core of Moodle; we were then recommended to pursue a patch submission via the plugin database.

All in all, I wish I could have given more time to the project. My other curricular and co-curricular obligations became overwhelming and I could not contribute fully. I feel that, had I been more available, we could have come farther, as a team. Nonetheless, this was a new experience for me, and I did learn first-hand about working with an established open source community, and with a team of developers seeking to contribute to it.

Appendix A. Scope Document for Moodle DB2 Conversion Project

Version 1.1

Prepared by Marc Graham, Paul Nguyen, Tony Love, Andy Huang

CECS 491, Fall 2009, Dr. Alvaro Monge

September 30, 2009

Table of Contents

Revision History

1. Project Requirements

1.1. Background

1.2. Project Goal

1.3. Project Objectives and Success Criteria

2. Scope and Limitations

2.1. Scope

2.2. Limitations and Exclusions

2.3. Project Implementation Plan

2.4. Project Change Request Handling

Revision History

Name	Date	Reason For Changes	Version
Marc Graham	09/30/09	Level of testing target for ADO-DB set to B	1.1

1. Project Requirements

This project entails moving Moodle's support for IBM DB2 from the theoretical to the actual in the 1.9.x development stream. This will include moving the testing status for IBM DB2's native PHP driver in the ADO-DB project from level C (user contributed and experimental) to level B (testing and use able).

1.1. Background

Moodle, the open source course management system, is used and well regarded by many organizations. Moodle's design has been and continues to be centered around the open source MySQL database product. Some alternative database systems are supported by the Moodle installer such as Postgres, Oracle 8i, and MSSQL, while others are considered supported but un-

implemented. IBM's DB2 database server is one such supported but un-implemented product. Certainly this lack of support has not hindered Moodle's growth and utilization in the past. As long as an organization was flexible enough to be able to deploy one of the installer-supported database platforms Moodle could easily be deployed. But this does limit the "market" for Moodle to relatively small organizations or organizations that have a supported database system. In the case of large institutions, it is often not possible to add a new database platform just to support one application regardless of that application's merits.

1.2. Project Goal

What this project will address is the lack of choice in selecting IBM's DB2 as a database platform for which Moodle can be installed. It is believed that in providing this functionality for current and future users of Moodle we will be able to open up acceptance of Moodle where it might not have been possible otherwise. We feel that the community of Moodle users should not be limited in their choice of database servers and should be able to take advantage of the IBM DB2 Express-C product. We also feel that larger institutions that already have an IBM DB2 infrastructure should not be dissuaded from using Moodle due to lack of support for their database platform.

1.3. Project Objectives and Success Criteria

The objective of this project is to make the selection of IBM DB2 as a database server for Moodle as easy as selecting MySQL. Success will be determined by four criteria:

1. Performing a clean install of Moodle with an IBM DB2 Express-C database server and verifying that the same functionality is attained compared to an installation with MySQL.
2. All Moodle source file modifications that implement #1 are submitted and accepted by the Moodle development community to be included in the current 1.9.x builds.
3. ADO-DB source file modifications that implement #1 are submitted and accepted by the ADO-DB community and included in subsequent releases.
4. The testing status of the ADO-DB IBM DB2 native driver is elevated to level B. A best effort attempt will be made go get ADO-DB testing level elevated to A.

2. Scope and Limitations

2.1. Scope

This project will concern itself solely with issues involved in enabling current and future users of Moodle 1.9.x to enjoy the use of IBM DB2 Express-C as their database server of choice when deploying. This will include (in equal priority):

1. Modification of Moodle PHP source files while strictly adhering to Moodle development guidelines.

2. Thorough unit testing as well as limited operational testing of Moodle functionality with IBM DB2 Express-C on Linux (version TBD).
3. Complete Moodle installer support for IBM DB2 Express-C.
4. Resolution of any ADO-DB related failures with fixes submitted to the ADO-DB community.
5. Generation of documentation chronicling issues and design decisions related to this work.
6. Submission of all modified source files to the Moodle developer community and ensuring acceptance.
7. Documentation for installers/admins as needed to ensure repeatable installation success with IBM DB2 Express-C.
8. Utilization of a source code repository and good management practice (TBD) to be followed to ensure code integrity.

2.2. Limitations and Exclusions

1. This project will address Moodle 1.9.5+ and specifically excludes any considerations for the upcoming Moodle 2.0 release.
2. Localization issues for Moodle relating to this enabling work will not be addressed. Where needed, we will provide IBM DB2 related message strings in United States English. There will be no attempt to translate any message strings to any other Moodle supported language.
3. Any functionality or enhancements to Moodle or ADO-DB that do not directly relate to enabling the use of IBM DB2 Express-C will not be addressed.
4. No modifications to IBM DB2 PHP native driver (.dll or .so) code will be attempted by this team.
5. This project will end December 16, 2009.

2.3. Project Implementation Plan

This project has two possible ways of proceeding that depend upon the ability to migrate an existing Moodle database to IBM DB2 Express-C. This work is on-going at the time of this writing (September 26th). Decision date on this issue is no later than October 3, 2009. If no usable migration occurs by the decision date, plan B will be executed.

Plan A - Make it work, then fix the installer. (Preferred plan.)

Phase 1: Begins no later than October 3, 2009 and ends by November 6, 2009.

Using a database migrated to IBM DB2 Express-C from an installed and functioning Moodle instance, a copy of the migrated Moodle instance will be connected to it. The included Moodle unit tests will then be executed and any errors generated will be analyzed and repaired.

This phase is complete when all Moodle unit tests pass using an IBM DB2 Express-C database as well as an instance running on MySQL using our modified files. Limited operational testing will be performed, as time allows.

Upon completion all files modified will be submitted back to the Moodle project.

Phase 2: Begins no later than November 7, 2009 to end by December 4, 2009.

Installer functionality will be enhanced to ensure Moodle can be consistently installed with IBM DB2 Express-c. This will include verifying that required XMLDB functionality is on par with an instance of Moodle running MySQL. Operational testing will continue in this phase.

This phase is complete when the installer consistently configures Moodle to use IBM DB2 Express-c and MySQL on a Linux (version TBD) platform and all required functionality is on par with a comparative MySQL implementation. While it is not reasonable to expect that every single Moodle function can be tested in the amount of time allotted to this project, every effort will be made to cover as much as is practical.

Upon completion all files modified will be submitted back to the Moodle project.

Phase 3: Begins no later than December 5, 2009 to end by December 16, 2009.

Clean up and documentation roll-up phase. Any unresolved issues will be documented and handed off to team members interested in working on the project after December 16, 2009. It is anticipated that post-December 16 work will be limited to ensuring submitted code is accepted.

Plan B - Debug the installer, then verify functionality.

Phase 1: Begins no later than October 3, 2009 and ends by November 6, 2009.

The Moodle installer and supporting files will be modified to install on an IBM DB2 Express-C instance. Resulting install failures will be analyzed and repaired until the installer completes successfully.

This phase is complete when Moodle will consistently install on both IBM DB2 Express-C and MySQL using our modified files.

Phase 2: Begins no later than November 7, 2009 to end by December 4, 2009.

The Moodle unit tests will be run and all failures analyzed and repaired. Operational testing will begin. XMLDB functionality will be verified.

This phase is complete when all unit tests pass and required functionality is determined to be on par with a comparative MySQL implementation.

Upon completion all modified source files from phase 1 and phase 2 will be submitted to the Moodle developer community.

Phase 3: Begins no later than December 5, 2009 to end by December 16, 2009.

Clean up and documentation roll-up phase. Any unresolved issues will be documented and handed off to team members interested in working on the project after December 16, 2009. It is anticipated that post-December 16 work will be limited to ensuring submitted code is accepted.

Note: These phases may overlap.

2.4. Project Change Request Handling

Due to the nature of this project and the timeline that we are working under, change requests will not be considered.

Appendix B. Project Links

Core Project Links

Google Group: http://groups.google.com/group/moodle_db2ExpressC/

BitBucket: <http://www.bitbucket.org/geek745/moodle-db2>

Timeline: <http://spreadsheets.google.com/ccc?key=0Aopb2SfFG8oDdHlZWF9EWkp3S1p2RVJNdEJvdjBsMFE&hl=en>

Moodle Test instance on AWS: <http://www.learn2.org/moodle>

Project Documentation

Oct 21. Project scope update.

http://docs.google.com/present/view?id=dhdjq6tr_3dtk79pdq

December 5. Final Report Outline:

<http://docs.google.com/Doc?docid=0AWjCW7rbcLwRZHFucWZxNI8zNTRzN24zcG5j&hl=en>

Reference Material

Moodle: <http://www.moodle.org>

ADODB: <http://adodb.sourceforge.net/>

Moodle Test Suite: <http://docs.moodle.org/en/Development:Tests>

Encryption in the US Government

Paul M. Nguyen
CECS-478, B. Englert
May 10, 2010

Table of Contents

Acknowledgments.....	2
Introduction.....	3
Common Threads.....	4
Department of State.....	5
Department of the Treasury.....	6
Department of Defense.....	8
Department of Justice.....	9
Department of the Interior.....	10
Department of Agriculture.....	10
Department of Commerce.....	10
Department of Labor.....	11
Department of Health and Human Services.....	11
Department of Housing and Urban Development.....	12
Department of Transportation.....	12
Department of Energy.....	13
Department of Education.....	13
Department of Veterans Affairs.....	14
Department of Homeland Security.....	14
Conclusion.....	16

Acknowledgments

I would like to thank Dr. Burkhard Englert for the opportunity to pursue this project. It has been quite eye-opening.

I would also like to thank Maria Santella, a colleague from class who performed half of the preliminary research for this project. Without the headway she established, this report would have been completed under much greater stress.

Introduction

The Federal Government of the United States of America is composed of 15 cabinet departments that execute the many and varied tasks with which the Federal Government is charged. Each department requires substantial information to operate, including the information about each client of the department, with respect to the department's portion of government operations, and the information required to organize the department's resources internally. This vast information, which has become increasingly dependent upon computer systems since the passage of the Paperwork Reduction Act of 1980 and its 1995 revision, must always be kept secure.

In the past, in order to ensure the confidentiality, integrity, and availability of physical documents, it was only necessary that documents be kept sufficiently available, that the place of storage guaranteed that only authorized persons accessed each document, and that their contents were not modified, except by authorized persons. The technologies that supported these measures were primitive, and only served to accelerate the mechanical execution of each. More advanced safes kept documents better protected against theft and access by unauthorized persons; higher fidelity duplication equipment allowed for more sophisticated verification processes to guarantee integrity; and advanced transportation technology helped to keep documents available, even at great distances.

The need has arisen recently, however, to maintain records via computer systems, and the analog to each component of the "old way" is now a bit incongruous. Records stored in a relational database, for example, are stored in fragments located in various files on a hard disk that represent that data. Relational databases improve availability while compromising the ease with which confidentiality and integrity may be maintained. Because of the nature of the storage

of the data on disk with respect to files (the natural container for data on disk), the relational database introduces a host of problems for guaranteeing security. With the advent of electronic messaging, including email, text messaging (SMS), wireless network access, and smart phones, yet another dimension of complexity is introduced. That is, that data is not guaranteed to remain within protected computer systems and data storage facilities; it must be protected in transit as well as at rest. In order to ensure security at each of these steps, a number of technologies and policies have been developed and prescribed for use by Federal Government agencies, which we will now explore.

Common Threads

There are several common threads that run throughout the discussion that follows. First, there are agencies within the Federal Government whose task it is to determine the standards of security with respect to information-bearing computer systems. The most notable are the National Institute of Standards and Technology (NIST), and the National Security Agency (NSA). Jointly, these two groups develop and certify technology used to secure the nation's information and infrastructures. Among the cryptographic standards established are the symmetric encryption algorithm DES (Defense Encryption Standard), the asymmetric successor, AES (Advanced Encryption Standard), PGP (Pretty Good Privacy), and X.509 digital certificates. The NIST has also approved a series of hashing algorithms for use, primarily the Secure Hash Algorithm (SHA) family. Each of these technologies was chosen for its resilience to compromise, but that security inevitably depends on each and every user's resolve to maintain it.

Nationally, there is a Federal Chief Information Officer (CIO), whose office, within the Office of Management and Budget¹, evaluates policies, practices, and technology to determine how to most effectively and securely operate the Federal Government. The OMB released a

1 E-Government Act of 2002. Public Law 107-347. 116 STAT. 2899

memorandum in 2007 to all departments, instructing them to develop policies within the following four months that detailed how to handle personally-identifiable information (PII) and how to proceed with notification in the event of a breach.² Several Cabinet-level departments also contain a CIO or Chief Information Security Officer (CISO).

There are two primary kinds of information that the various departments seek to protect. The first is called “vital information” and refers to information necessary to maintain minimum operation of the Department, and includes historical records for the department. The other is called personally-identifiable information (PII) and refers to personal information of US citizens, including anything that might, in possible combination with other information, locate or identify a person without their knowledge and consent.

Department of State

The State Department, the Secretary of which is third in line to succeed the President, is generally responsible for diplomacy and interfacing with foreign political bodies. The State Department is also responsible for regulating passports and visas. The Department of State declared two important policies governing the use and security of information within it – one regarding the Internet-based SharePoint service, and the other regarding US passports containing electronics.

The Department of State document describing the secure use of Microsoft Office SharePoint Services demonstrates the priority placed on information security within this department.³ The DOSSSI is meant to provide the Department with the ability to collaborate internally, and for field agents to access, both for reading and writing, information concerning the department and its customers, the citizens of the country. SharePoint's integration with the

2 Office of Management and Budget. M-07-16. 22 May 2007.

3 Grafeld, M. P. Department of State SharePoint Server – Internet (DOSSSI). Retrieved May 6, 2010 from <<http://www.state.gov/documents/organization/139731.pdf>>

rest of the Microsoft Office productivity suite made it an ideal choice for functionality, but at the same time, raised security concerns as to the workflows that would keep PII private, as needed. The document on DOSSSI specifies that only authorized users may input or facilitate the input of PII into the system, and that once it is received, that information is not to be released to anyone other than the US Secret Service, and by exception, other required persons. The policy further states that PII shall not be transmitted to other systems for processing in an automated fashion at all. This system is rare in that, for the triad of security concerns, availability of the secured data is irrelevant; confidentiality and integrity become primary.

US Passports began bearing radio frequency identification (RFID) circuitry to expedite the process of scanning passports and to provide a measure of further integrity verification. By August of 2007, all new passports issued in the US were of this new type. The e-passports, as they are called, sport passive RFID technology that only responds to a reader within range and requires no power source of its own. To protect against “skimming,” the process of gaining the information stored on the chip in an aggregate manner (such as sitting in a public place in which passports are carried), the Department of State recommends that carriers of e-passports should do so in electromagnetically-shielded containers that prevent such behavior. The e-passports are further protected internally using a public key infrastructure (PKI) derivative that certifies authenticity and guards the actual information on the chip.⁴

Department of the Treasury

In the beginning, departments turned to DES, and the Treasury was one of those departments. In a 1992 directive, the electronic funds transfers for the department were required

⁴ The U.S. Electronic Passport Frequently Asked Questions. Retrieved May 10, 2010 from <http://travel.state.gov/passport/eppt/eppt_2788.html>

to be encrypted using DES.⁵ The Directive cites the authority of the NIST and FIPS 140 in its issuance. Years later, in a speech to the Computer and Communications Industry Association in mid-2000, remarks were made about having lifted export controls on encryption tools.⁶ This change in regulation helps fuel the production of stronger encryption techniques and promotes the growth of security nationwide.

The Department of the Treasury's Office of the Inspector General released a Policy Directive on May 30, 2007 that detailed office policies for handling PII and for computer security in general.⁷ In this directive, OIG employees are directed to protect against unauthorized access of department computing resources, including hand-held devices and removable media. Encryption is required on as many devices as is feasible without unreasonably obstructing availability. Activities including locking one's computer in software, ending one's session through the "Log off" feature, and shutting down one's computer overnight are some of the physical precautions mentioned in the directive. It is also critical that information in transit, either via the network or removable media, be encrypted using WinZip or built-in technology on Blackberry hand-held devices. Finally, rules concerning password strength are prescribed, including the use of hard-to-guess information and punctuation marks.

As part of an audit for Fiscal Year 2009,⁸ the OIG released a report that described shortfalls in the implementation of minimum configuration requirements documents concerning computing resources that would provide sufficient security for the Office's operations.

5 Electronic Funds and Securities Transfer Policy Message Authentication and Enhanced Security. Treasury Directive 16-02. 21 December 1992.

6 Eizenstat, S. E. Department of the Treasury. LS-677. 5 June 2000.

7 OIG Information and Physical Security. Policy Directive 710-02. 30 May 2007.

8 Management Letter for Fiscal Year 2009 Audit of the Department of the Treasury's Financial Statements. OIG-10-035. 15 December 2009.

Department of Defense

The Department of Defense (DoD) released a memo in mid-2007 detailing handling procedures for sensitive data present on mobile devices.⁹ This policy document mandates the use of encryption for all kinds of data on such devices as laptops, mobile phones, and removable media used to transport information outside of DoD facilities. Particularly interesting is a statement in the March 2008 clarification document¹⁰ that Microsoft's Encrypted Filesystem (EFS) solution does not satisfy FIPS 140-2 and is therefore unsuitable for protecting sensitive data at rest on portable devices or media.

The Navy Information Security Program outlines practices and training methods to be used in keeping information secure. Its 289-page manual,¹¹ published in 2006, prescribes a complete program, including identification of sensitive information, authorization of personnel to access sensitive information, and training practices for instructing personnel on the handling of sensitive information. Finally, the manual addresses what to do in the event that data is lost.

The Department of Defense maintains a sophisticated agency responsible for all these matters, the Defense Information Systems Agency (DISA). DISA is responsible for providing the communications infrastructure needed to connect military personnel in every situation, from the Whitehouse to the front lines, and to provide for completely secure communications, addressing availability, confidentiality, and integrity of those communications.

The DoD also set up a system recently, called SIPRNet,¹² which provides a hardware token for PKI authentication via X.509 certificates. This system was scheduled to become active

9 Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media. Department of Defense. 3 July 2007.

10 Frequently Asked Questions: Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media. Department of Defense. 19 March 2008.

11 Secretary of the Navy. Department of the Navy Information Security Program. SECNAV M-5510.36. June 2006.

12 SIPRNet. Department of Defense Information Assurance Support Environment. Retrieved 10 May 2010 from <<http://iase.disa.mil/pki/index.html>>

only months ago, in Spring 2010. Such a system provides greater reliability for authentication since the user must possess a physical object that contains digital information and thus provides a method by which physical presence for authentication might be validated in the digital realm, where authentication and identity are typically less reliable.

Department of Justice

The Department of Justice (DoJ) released a report in early 2010 describing the use of encryption on laptop computers within the department.¹³ In the Criminal Division, sections were identified in the audit that did not comply with baseline configurations' security rules in that they were not encrypted. The report reiterated the Department policies that mandate encryption of computing resources that process information relating to criminal cases, and enumerated the offending items. This type of oversight is precisely what weakens the overall strength of the Department's computer systems against possible attacks; the more weaker links exist, the more likely it is that a breach of security can occur.

The Department of Justice also released a report¹⁴ about voice-over-Internet-protocol (VoIP) technology, stating that due to poor interoperability, it should not be adopted for official business. It also makes clear that until such time as the Department can develop standards governing its secure use, it should not be used. This restricts availability of the communication channel while providing for confidentiality and integrity of data that is kept out of the known insecure channel. The DoJ further addressed the issue of radio communications between public safety personnel, that they are vulnerable to eavesdropping and should, therefore, be encrypted

13 The criminal division's laptop computer encryption program and practices. Department of Justice Office of the Inspector General. Audit Report 10-23. March 2010.

14 Voice over Internet Protocol. Department of Justice Office of Justice Programs National Institute of Justice. NCJ 217864. May 2007.

using AES or DES.¹⁵

Department of the Interior

The Department of the Interior (DoI) maintains an extensive manual of operating procedures that contain provisions for the assurance of important information, both critical to the daily operation of the department and critical for protecting the nation's natural resources. Within the Manual, the Chief Information Officer is charged with maintaining the viability of computer systems and the security of the data they contain.¹⁶ Specifically, physical security, firewalls, and encryption are listed as means of ensuring the integrity and confidentiality of data as it passes through untrusted zones.

Department of Agriculture

The Department of Agriculture (USDA) established the Chief Information Officer position in response to the e-Government Act of 2002. Under this office and the associated Information Security Program of the USDA, standards are to be enforced that concern encryption, handling of unclassified sensitive information, such as PII, and generally enforce stringent access control and short-termed retention rules. USDA requires compliance with FIPS 140-2 (AES and Triple-DES) and permits the use of IPsec, VPN, SSL, and PKI technologies to secure communications, among others.¹⁷

Department of Commerce

The Department of Commerce documentation on encryption primarily concerns the

15 Voice Encryption for Radios. Department of Justice Office of Justice Programs National Institute of Justice. 14 November 2007. Retrieved 10 May 2010 from <<http://www.ojp.gov/nij/topics/technology/communication/voice-encryption.htm>>

16 Information Technology Security Program. Department of the Interior Department Manual. 375 DM 19. 15 April 2002

17 Encryption Security Standards. US Department of Agriculture Office of the Chief Information Officer Department Manual. DM3530-005.

export of encryption tools, under the Bureau of Industry and Security.¹⁸ Essentially, the export controls could seek to guarantee that encryption used within the US is stronger than what is used by the rest of the world, both so that secrets may be hidden within that others cannot determine, and that experts in the US would be able to decode material found elsewhere around the world.¹⁹

Department of Labor

The Department of Labor does not lay claim to any internal policies beside those issued from the core of the Federal Government. It refers to the e-Government Act of 2002 and its FISMA provision, the Freedom of Information Act as amended in 1996, and the Paperwork Reduction Act of 1995.

Department of Health and Human Services

The Centers for Disease Control and Prevention (CDC), an agency of the Department of Health and Human Services (DHHS), stipulates²⁰ that for testing of HIV and AIDS, data concerning patients must be encrypted. It provides for the high-grade destruction of physical documents via cross-cut shredding, and requires that computer systems used to store data should be few and well-protected. AES and VPN are required for transmissions over external networks when such sensitive data must be stored at a remote location. Like other departments, mobile devices require encryption, and removable media used to transport sensitive information must also be encrypted.

18 Notification Requirements for "Publicly Available" Encryption Source Code. Department of Commerce Bureau of Industry and Security. Retrieved 10 May 2010 from <<http://www.bis.doc.gov/encryption/pubavailencsourcecodenotify.html>>

19 Commercial encryption export controls. Department of Commerce Bureau of Industry and Security. Retrieved 10 May 2010 from <<http://www.bis.doc.gov/encryption>>

20 Centers for Disease Control and Prevention and Council of State and Territorial Epidemiologists. Technical Guidance for HIV/AIDS Surveillance Programs, Volume III: Security and Confidentiality Guidelines. Atlanta, Georgia: Centers for Disease Control and Prevention; 2006.

Department of Housing and Urban Development

The Department of Housing and Urban Development (HUD) lays out very clear guidelines²¹ regarding the protection of key information, and specifically encourages the use of AES rather than triple-DES, due to the fact that triple-DES will soon be phased out. It also makes careful note of physical security precautions that must be taken – both that hard copies of sensitive information remain within the immediate personal possession of those authorized to carry it, and that they be satisfactorily destroyed once they are no longer needed.

The HUD Handbook also specifies security concerns that must be addressed within the Department.²² It addresses authentication via PKI and hardware tokens, and identification and authentication of network-attached devices, and does so referring directly to the NIST requirements pertaining to each area. The Handbook further addresses the need for employees to conduct official business only on official machines, and to conduct no personal business on official machines, in order that cross-contamination of data is minimized.

Department of Transportation

The US Department of Transportation (DOT) conducted a Privacy Impact Assessment in 2007 and, in the report,²³ carefully described that PII must be handled carefully by a 3-tier model of users and administrators, including such provisions as password complexity rules, restricted physical access, and software access controls that enforce the policy of least privilege.

The Society of Automotive Engineers (SAE) cooperated with DOT to formulate the ITS Data Bus, which is a serial communication interconnection protocol and hardware design that facilitates communication between electronic components within an automobile. Together with

21 Safeguarding Privacy-Protected and Sensitive Data. US Department of Housing and Urban Development Office of Public Housing and Voucher Programs. 10 July 2006.

22 HUD Handbook 2400.25 REV-2, CHG-1.

23 Privacy Impact Assessment: Enterprise Support Systems. US Department of Transportation. Retrieved 10 May 2010 from <http://www.dot.gov/pia/ost_ess.htm>

the ITS Data Bus, a security document²⁴ was released. In this document, SAE refers to the basic Triple-DES and AES standards and also makes note that with a common data bus, it is necessary for components to encrypt their communications with other similar components. This is a unique application of encryption thus far because it does not involve users' or clients' PII directly, but facilitates the integrity-rich operation of machinery this department is charged to ensure.

DOT is also responsible for air traffic control, a direct responsibility of the Federal Aviation Administration (FAA). A standards document on communication security²⁵ directs NAS sites to use a host of encryption technologies to secure their communications with one another, including using PKI and PGP, SSL, SSH, and IPsec with VPN connections. The significance of this encryption practice is also somewhat secondary to the PII requirements seen in other departments. In this case, the goal is the protection of data required for the Department, itself, to function – to keep airplanes in the air and keep them from colliding.

Department of Energy

The Department of Energy takes into account all of the federal regulations concerning the security of vital information for each department. Evidence of this is found via several federal documents' copies found within the DOE CIO website, particularly concerning Records Management.

Department of Education

The Department of Education maintains similar baselines to the other departments with respect to encryption of data at rest and in transit; FIPS 140-2 (AES and PKI) is required for data at rest, and VPN links are required for transit of sensitive information, according to the

24 SAE J1760 - ITS Data Bus Data Security Services. US Department of Transportation. 1 October 2005. Retrieved 10 May 2010 from <http://www.standards.its.dot.gov/fact_sheet.asp?f=48>

25 National Airspace System (NAS) Communications Security Protocols and Mechanisms. Federal Aviation Administration. FAA-STD-045A. 11 March 2005.

handbook.²⁶ The maintenance of this baseline seems to be the uniform standard of the government, both in theory and in practice.

The No Child Left Behind program released some guidelines encouraging the adherence to federal policies concerning encryption of PII, including education-related records.²⁷

Department of Veterans Affairs

The VA Office of Information Protection and Risk Management is responsible for publishing standards and workflows that create a secure environment for working with sensitive data. In late 2006, the VA made a move to encrypt all mobile computers and removable media.²⁸ The VA Handbook makes it clear that no person is to introduce foreign systems into the network.²⁹ Finally, the VA maintains a PKI program³⁰ whose purpose is to keep the VA current with NIST standards changes and to make sure that feasible workflows exist to incorporate the strength of PKI security into VA networks.

Department of Homeland Security

The Department of Homeland Security (DHS) is arguably one of the more important departments to secure; it appears last in this list because it is the most recently-created. The Office of the Inspector General for DHS issued a report in October 2009³¹ stating that radio equipment used by border patrol personnel was not sufficiently secure; legacy equipment was used that was incompatible with radio encryption standards mandated for the rest of the

26 Handbook OCIO-15. Department of Education Administrative Communications System. 30 March 2007.

27 Improving Data Quality for Title I Standards, Assessments, and Accountability Reporting. US Department of Education Office of Elementary and Secondary Education. April 2006. Retrieved 10 May 2010 from <<http://www2.ed.gov/policy/elsec/guid/standardsassessment/nclbdataguidance.pdf>>

28 VA Secretary Unveils Data Security Encryption Program. US Department of Veterans Affairs. 14 August 2006.

29 VA Handbook 6500. 18 September 2007.

30 <http://www4.va.gov/proj/vapki/>

31 Review of the U.S. Customs and Border Protection Expenditure Plans for the American Recovery and Reinvestment Act of 2009. US Department of Homeland Security Office of the Inspector General. October 2009. OIG-10-05.

department. The Federal Emergency Management Agency (FEMA), for example, requires AES-encrypted radios for their use.³² DHS even released a primer describing encryption technology so that architects of control systems could depend on the supporting groundwork of encryption when developing further tools.³³ The TSA, another sub-agency concerned primarily with keeping travelers and critical infrastructure safe, published some material³⁴ reiterating the importance of encryption to protect personal information.

32 FEMA 508-2, Incident Management Resources. U.S. Department of Homeland Security Federal Emergency Management Agency. 20 March 2006. Retrieved 10 May 2010 from <http://www.fema.gov/txt/emergency/nims/incident_mgmt.txt>

33 Department of Homeland Security: Control Systems Communications Encryption Primer. Retrieved 10 May 2010. <http://www.us-cert.gov/control_systems/pdf/Encryption%20Primer%20121109.pdf>

34 Handling Sensitive Personally Identifiable Information. TSA Management Directive No. 3700.4. Transportation Security Administration.

Conclusion

Computer security is a constantly moving target – developments in attacks cause defenses to mature, and vice versa. It seems that the Federal Government of the United States of America has exerted significant effort to keep pace with the changes, lagging about 2-4 years behind the curve. Although the technologies that are presently being installed across the entire government are still quite viable, their first introduction to the market was a solid decade before the federal government as a whole absorbed the new technology.

It is fascinating how uniform some departments' regulations and procedures are concerning encryption of vital data – they took the textbook version of the e-Government Act and the Paperwork Reduction Act along with the NIST regulations and declared them to be relevant for whatever type of data was managed by that department. By the same token, it is equally fascinating how different some departments' regulations are. It is also significant to mention, at this time, that some departments may not have made their regulations concerning encryption readily available on the Internet, making it difficult for the public to ascertain the policies in place for these departments or agencies.

The most surprising application was the application of AES encryption to radio communications. This encryption scheme generates a private and public key that can then be used (provided the Man-in-the-Middle attack is prevented) in two-way communication with another client radio on the same network. It is absolutely necessary that emergency response personnel be able to communicate easily and securely from the field and a central location, and real-time encryption in PKI is the perfect solution.

All in all, the Federal Government is aware of technologies and workflow considerations that must be made to ensure security, and it is only a matter of compliance to achieve it.

Conclusion

I thoroughly enjoyed the opportunities to work on Open Source software and to study Computer Security in the applications of encryption within the US Government. It is experiences such as these that provide students contact with the real world as relevant to their careers that really make education meaningful and form students into the thinkers that will proceed to make a profound impact on the world.

The classroom experience here at the CSU is one that combines solid theory with meaningful practice, and these senior experiences are just the crowning finish to a continuous practicum in Engineering. Labs associated with each lecture immediately put concepts to use solving the classic problems in each field and quickly expand to engage a more creative approach to the material. Trivial examples give way to advanced inquiry and cookbook-style problems lead to open-ended ones requiring creativity, planning, and resolve to execute successfully.

I am endlessly grateful for the experiences I have gained as a part of the University Honors Program, the Computer Engineering and Computer Science Department and the College of Engineering, and all of California State University, Long Beach. These experiences are real, with tangible benefits and outcomes, and have well-prepared me for a dynamic and rewarding future. Now, let the theory hit the highway!