Security Vulnerabilities in the open source Moodle eLearning System

Colton Floyd, Tyler Schultz and Steven Fulton, US Air Force Academy

Abstract – Moodle eLearning System is well known as a free web application e-learning platform used in many schools as a way to allow on-line student interaction. Institutions use Moodle for its flexibility, adaptability and ease of use. Moodle has an installation base of tens of thousands of institutions with millions of student users. Our institution uses Moodle for admission of timed on-line quizzes taken outside the classroom as well as a vehicle for students to submit homework assignments. This paper outlines well-known vulnerabilities for Moodle v. 1.9 and attempts to exploit these vulnerabilities as well as identify new vulnerabilities in Moodle v. 2.1. Surprisingly, many of these known vulnerabilities continue to exist following their identification suggesting possible vulnerabilities which can be exploited by students. Based on these known vulnerabilities, we made recommendations on how to correct these problems in future Moodle versions and individual Moodle configurations.

I. Introduction

With internet connectivity becoming more affordable and prevalent education and training has evolved to the point that learning occurs not just in a classroom but online as well. Today eLearning, "computer-based training", and "virtual instruction" are often done in conjunction with classroom instruction or exclusively online. The desire for online learning and teaching tools is not just for educational institutions, but corporations as well. Typically schools and corporations spend large sums of money to develop custom training modules or obtain commercial closed-source web-based course management suites such as BlackBoard or WebCT. A new alternative released in 2001, Modular Object-Oriented Dynamic Learning System (MOODLE), has quickly become one of the most popular and successful open source eLearning suites.

II. The Need for eLearning

Virtual universities, education portals, complete online courses, and electronic course supplements are some examples of the digital learning environments that have recently developed [1]. In order to maintain a competitive edge, many universities have developed online courses or offer complete online degrees in addition to the traditional classroom environments. Innovative solutions and approaches to eLearning are not replacing traditional learning processes, but enhance and extend it. Modern student populations are beginning to expect some form of an eLearning option from traditional institutions as evidenced by the increased demand for it [2]. Corporate training is also provided by many organizations in some form of online course. Education portals are often used for collaboration or other social activities alongside classroom or online instruction, but are not directly instructional or considered course material. Class supplements include quizzes, tests, web-based lessons, or homework assignments. Features of eLearning suites often include electronic file/assignment submission, grade keeping, calendars, forums, bulletin/notification systems, chat rooms, and some form of a classroom "portal" or web page in addition to the common assessment features and web-based lessons.

III. Moodle Functionality

Moodle offers many of the common eLearning features through a collaborative interaction model of eLearning. The intent is that students will do additional work outside of class through self-interests and exploratory learning using Moodle as the delivery mechanism for the standalone or "hybrid" (online and classroom based) instruction [3]. Moodle provides templates for various modules, with a module being similar to add-ons in various modern web browsers. The system allows for various formats, such as plaintext, HTML, common video/audio, and even PowerPoint or Flash presentations to be used within the modules and integrates with site data to form searchable glossaries. Testing modules provide multiple guess, fill-ins, multi-select, true/false, matching, short answer, essay, and matching question types in addition to administration options such as time limits, multiple attempts, deadlines, automatic grading, and/or password protection [3]. The "Workshop module" is similar to a peer assessment tool, and the "essay module" allows for teachers to provide direct and specific feedback to students. Learning modules can be step-by-step completion or advance upon mastery; learning can be interactive through the use of forums, chat rooms, and Wiki modules with multiple configurations and functionality so that the teacher can have a large amount of control over the collaborative environment. PayPal can even be used to charge fees for courses offered by Moodle environments [4].

ISBN 1-933510-95-1/$15.00

IV. Known Security Issues

Moodle, like any other software, is not perfectly secure for a variety of reasons. Moodle, for better or worse, is open source software. As with any piece of open source software, Moodle can be viewed, modified, and improved by all users. Vulnerabilities can be found early, but they can also be exploited before patches are available. Easily viewable code gives a hacker the advantage. New vulnerabilities, such as backdoors, can be added by malicious contributors or poor coding can result in vulnerability [5]. That being said, suggesting just because a system is closed source doesn't suggest that a system is more secure. Such closed systems can still be exploited. Open source software, however, can be modified, so users can disable or patch vulnerable their implementations or find another way to secure Moodle. Therefore, in our view the openness of the source code is both an advantage and disadvantage to the end user.

Other known risks specific to Moodle include authentication, availability, confidentiality, and integrity attacks. Session management in Moodle is not inherently secure; communications are not always done over SSL throughout the entire site, if at all, leading to possible session hijacking. Moodle does not always require users to re-authenticate due to session caching, and doesn't restrict access through URLs. Moodle's weak cryptographic storage (use of non-salted MD5 hashes for instance) and direct object references (files or database records in URLs) without security checks pose a risk [7][9], malicious files can be uploaded to Moodle and potentially executed or shared to other users, buffer overflows, cross site request forgery, SQL injection, and cross site scripting vulnerabilities also exist [7]. Furthermore, it is not fully known what vulnerabilities within Moodle exist. Is it possible for a student to view/change grades on assignments for themselves or others? Can administration options on quizzes be disabled allowing more time on an assessment than allowed, or can a student elevate their privileges to that of a teacher or administrator? Moodle doesn't handle username and password guessing well, as there is no built in mechanism to prevent brute forcing username and passwords[7].

V. Relevance of Risk

All of these security risks pose problems not only to end-users, but to institutions as well. Open source software often leaves end users and organizations without a place to "point their finger" when programs fail or when there are security and/or privacy breaches [1]. Moodle is not immune to these problems either. With 56,185 active sites and 46,343,749 end users, Moodle has a substantial user base at many institutions [8]. The impact of a severe breach could lead to lawsuits, loss of business, classes, or additional materials at one or more institutions, or result in a lack of confidence in Moodle and other eLearning suites as a viable platform for digital education [1].

VI. System Configuration

Our security review used a series of three virtual machines and a production server. *Table 1: Moodle Configuration* describes the specific test environment used during the investigation of Moodle.

Table 1 - Moodle Configuration

Machine	Configuration	Usage
Primary VM	• Moodle Version 2.1 (1 Aug 2011 Build) • MySql Server Community Edition 5.1.41 • Ubuntu 10.04 • Apache 2.2.14 • PHP 5.3.2 • Username/Password ("Default") authentication system for Moodle • Default system settings	*This machine acted as the Moodle Server in a Ubuntu Linux environment.*
Secondary VM	- Backtrack 5	Backtrack provides a series of tools available to provide vulnerability assessment of the Moodle System.
Third VM	- Windows Server 2008 R2 - IIS 7.5 with SSL - Moodle Version 2.1 (1 August 2011 Build) - PHP 5.3.2 - MySql Server Community Edition 5.5.16 Default system settings	- Provided the role of a Windows 2008 server with Moodle installed.
Production Server	- Jasig Central Authentication Service 3.4.2.1 - Smart Card (CAC/PKI) Environment Windows Active Directory	l Moodle Server

VII. System Testing

A series of tests were performed using the system configuration defined above. Each of these testing modules was performed using off the shelf Moodle version 2.1.

1. *Session Hijacking:*

   Using the standard Moodle installation over HTTP, an attacker could hijack a Moodle session by stealing only two cookies. By default, Moodle cookies are only set to "session". The "HTTPOnly" and "Secure" flags are not set on cookies, in particular when they

are sent over plain HTTP. This allows an attacker to obtain cookies either through cross site scripting or through HTTP traffic sniffing.

The two cookies that Moodle uses to identify an authenticated session are "MoodleSession" and "MOODLEID_". The first cookie is a 26 character (208 bit) UTF-8 string that is randomly generated at the start of a new session, and the second cookie is generated from the user's account name at login. By stealing these cookies, a session can be successfully hijacked. The following test cases were used to verify the results.

Test Case 1 – Attempting Hijacking on the Same Machine.

To see if session hijacking can be done, two separate browsers were opened. In Google Chrome, the sample student account was then authenticated and navigated to a syllabus page for a sample course. In Firefox, the browser was navigated to the homepage and not authenticated. Using the Google Chrome developer tools, the two cookie values were copied and the cookies in the Firefox browser were updated. Both browsers were refreshed, and both sessions were fully functional. The session was successfully hijacked.

Test Case 2 – "Attempting Hijacking over the Network

To validate that Moodle sessions could be hijacked over the network, a sample teacher session was authenticated on one virtual machine using Google Chrome while a separate virtual machine was running Wireshark on the Backtrack 5 virtual machine. An active session from login to course administration was conducted, viewed in plaintext throughout, and login credentials as well as session cookies were seen in the Wireshark capture. On the machine running Wireshark, cookies values were added to a Firefox browser running a fresh session, and the page refreshed. In a separate browser on a different IP address, the session was successfully hijacked. Examples of the Wireshark packet capture is provided and highlighted in *Figure 1: Wireshark Network Capture*. Details of the frame captured are provided in *Figure 2: Frame Capture*.

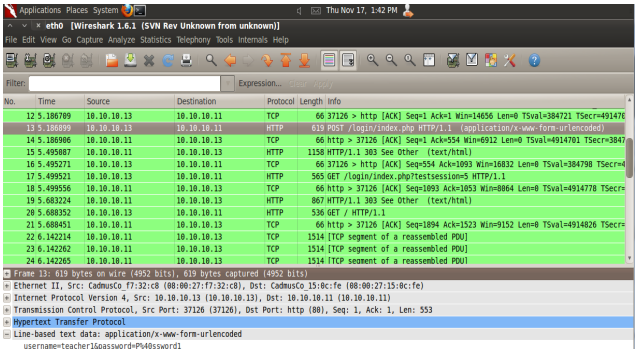

Figure 1- Wireshark Network Capture

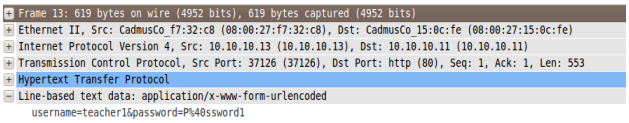

Figure 2 - Frame Capture

Test Case 3 – Live Server Test

At our university, the Moodle production environment authentication is done over SSL using smart cards to an open-source CAS server. The CAS server uses the certificates from the smart cards and LDAP to match the network account to a Moodle account. After authentication is done over SSL, communication is switched back to HTTP only and the session cookies are then transmitted to the client. These cookies are visible to network traffic sniffers. Using the data obtained from the HTTP session and repeating the procedure in Test Case 2, a session was successfully hijacked to another browser on another machine that is not joined to the domain and does not have any smart card credentials to pass to the Moodle server. *Figure 3: Pre-CAS Authentication* and *Figure 4: Return from CAS Authentication* outline these two configurations.

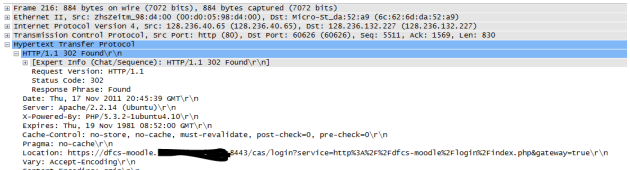

Figure 3 - Pre-CAS Authentication

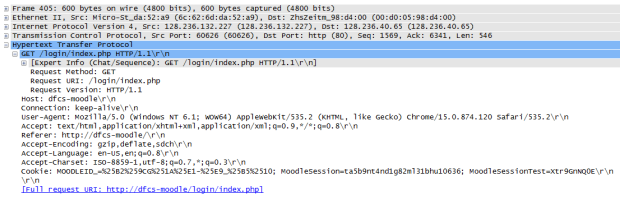

Figure 4 - Return from CAS Authentication

2. *SQL Injection:*
   Virtually every input field on common teacher and student account forms and any applicable URL parameters were tested for SQL injection vulnerabilities. No vulnerabilities were found on the areas searched.
3. *Cookie "Best Practices":*
   Moodle generates session ID's using a random string generation function. No noticeable pattern emerged, and session cookie values are 26 characters long, or 28 bits when encoded in UTF-8 (default). Moodle cookies over HTTP only connections are only marked as "session"; the "HTTPOnly" flag, which is used by browsers to prevent client side script access to cookies [10], is not marked. Cookies sent over SSL are also only marked as "session". The "secure" and "HTTPOnly" flags are not set, and therefore cookies sent over SSL are unencrypted and can be accessed by client side scripts.
4. *XSS Injection:*
   Overall, Moodle's has very good XSS protection capabilities. A similar approach to SQL injection testing was performed on all available inputs and URL parameters to determine if an XSS vulnerability exists. Not all possible pages or modules of Moodle were tested, but the wiki, syllabus, quiz, search, login, quiz, forum, course creation, user messaging system, blogs, blank page, and chat modules were tested. A vulnerability exists in the "URL" resource available to administrator accounts where a specially crafted URL can be created that displays XSS scripts. The following URL was added successfully:

```
evil.com/url=something&script=<scri
pt>alert('XSS SAYS HI!');</script>
```

   Figure 5: URL Resource Configuration Page and *Figure 6: URL Results* demonstrate the XXS vulnerability outlined.

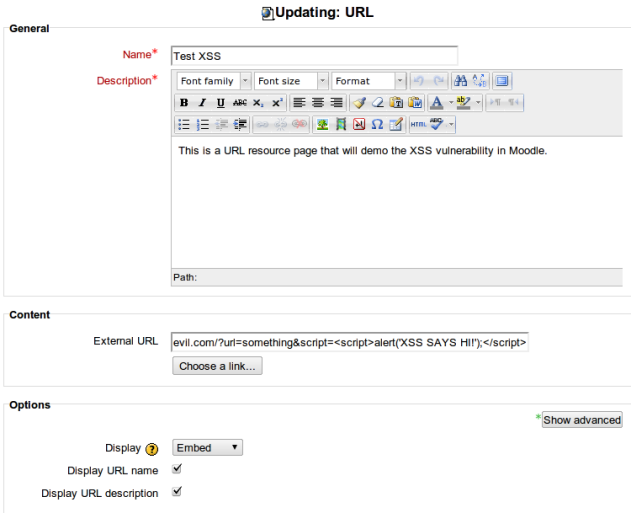

Figure 5 - URL Resource Configuration Page

After clicking the link "Test XSS":

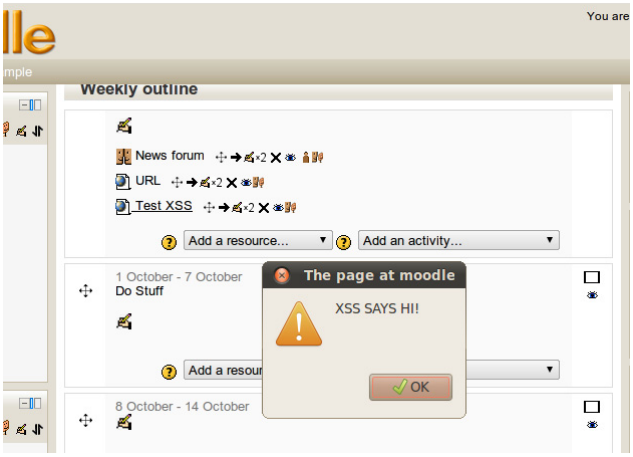

Figure 6 - URL Results

5. *Session Management Flaws:*
   The previously mentioned session hijacking vulnerability is one session management flaw. Moodle, although it does store IP addresses associated with a session, does not verify the session cookie and request IP against the database. It is possible for a single (identical/hijacked) session to continue simultaneously from two different IP addresses and browsers.

   Another session management flaw is account lockouts. Automated username and password guessing can be done easily as the server blocks logins after 10 invalid attempts. The invalid attempts are tracked only by the session cookie. If cookies are deleted automatically or never stored, it is possible to brute force a login known accounts. Moodle has no provision to lock user accounts completely after *x*

number of invalid attempts, and does not block an IP address from being able to authenticate after *x* number of invalid attempts.

6. *Quiz Engine Flaws*:
One feature of the Moodle quiz engine is the ability to put a time limit on quizzes. The system that enforces the time limit is a client side JavaScript function. Disabling JavaScript at the browser allows a student to bypass the time limit and still submit the quiz. Moodle does not enforce the time limit on the server side. It does track start and end times and reports the duration of each attempt to the teacher. This output is standard for the quiz engine, and there is no flag or other indication to the teacher that a student exceeded the time limit on the quiz. Teachers would have to verify that the duration for each attempt for every student and quiz to enforce the time limit. This flaw is demonstrated in *Figure 7: Pre-Quiz Dialog Showing Time Limit* and *Figure 8: Grade Results showing Time Taken*.

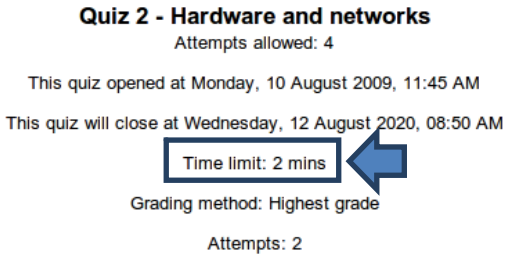

Figure 7 - Pre-Quiz Dialog Showing Time Limit

Figure 8 - Grade Results showing Time Taken

VIII. Summary of Findings

Table 2: Summary of Findings describes the vulnerabilities that we looked for in our examination of Moodle. Note that the issues found are identified as either *Found*, *Not Found* or *Some Found* (which suggests that some of the original issues still exist).

Table 2 - Summary of Findings

Vulnerability	Status
Session Hijacking	Found
SQL Injection	Not Found
Cookie "Best Practices"	Some
XSS Injection	Some
Session Management Flaw(s)	Found
Quiz Engine Flaw(s)	Found

IX. Recommendations

Our recommendations are split into two categories. The first is *Moodle Architecture Changes*, which provides proposals for changes to the Moodle system itself. The second category is *Implementation Recommendations* for organizations that use or are standing up Moodle servers.

A. Moodle Architecture Changes

Moodle's session management should be modified so that additional checks are placed on sessions. IP address, browser user agent, and other unique identifiers associated with a request could be used to verify that a session has not been hijacked. An option should be created to block simultaneous logins under the same account. In non-SSL environments, HttpOnly cookies should be set by default. For SSL, HttpOnly and Secure cookies should be set by default.

Logins for user accounts should be tracked by account, not by session. After a predetermined limit has been reached, the entire account should be locked for a given time interval. To prevent username and password brute forcing, an option to limit the number of invalid login attempts by IP address should also be made available.

Moodle's quiz engine should be modified to ensure that the time elapsed for timed quiz submissions is not greater than the time limit. This can be through "blocking" the submission or otherwise notifying the instructor of the discrepancy.

B. Implementation Recommendations

Moodle should be run completely under SSL if possible; if this is not possible, logins should be performed under SSL to prevent stealing of plaintext passwords. Cookies should be marked as HttpOnly and Secure. This can be achieved by navigating to "Site Administration" > "Security" > "HTTP Security" and checking "Use HTTPS for logins", "secure cookies only", and "Only http cookies". These changes prevent session hijacking and the viewing of all communication to the server, including

student answers, teacher's creation of assessments, and passwords provided at login.

X. Conclusion

Moodle is a popular open source learning management suite that provides a large suite of tools and modules for online education and training. It is arguably one of the most popular open source alternatives to commercial eLearning or Learning Management Systems such as Blackboard and WebCT. With a large user base and the ability to attack or compromise servers and web applications becoming easier and requiring less in depth knowledge, it is important that Moodle's security should be researched so that protections can be implemented and code patches or updates be released, not only for the benefit of Moodle and the institutions which implement it, but also for protection of student and teacher users.

XI. References

[1] Lakhan, Shaheen E., and Kavita Jhunjhunwala. "Open Source Software in Education."EDUCAUSE QUARTERLY 2 Nov. 2008: 33-40. Web.

[2] O'Neill, Kayte, Gurmak Singh, and John O'Donoghue. "Implementing ELearning Programmes for Higher Education: A Review of the Literature." Journal of Information Technology Education 3 (2004): 320-21. Web.

[3] Brandi, Klaus. "ARE YOU READY TO "MOODLE"?" Language Learning & Technology 9.2 (2005): 16-23. Web. 28 Aug. 2011.

[4] Stanford, Jeff. "In the Mood for Moodle." English Teaching Professional 54 (2008): 58-60. Web.

[5] Adkins, Sam S. "Wake-Up Call: Open Source LMS." American Society for Training & Development 2005. Web.

[6] Hoepman, Jaap-Henk, and Bart Jacobs. "Increased Security Through Open Source."Communications of the ACM 50.1 (2007): 79-83. Web.

[7] Kumar, Sheo, and Kamlesh Dutta. "Investigation on Security in LMS Moodle." International Journal of Information Technology and Knowledge Management 4.1 (2011): 233-38. Web.

[8] Moodle Trust. "Moodle Statistics." Moodle.org. Moodle Trust, 5 Sept. 2011. Web. 6 Sept. 2011. <http://moodle.org/stats/>.

[9] Stapić, Zlatko, Tihomir Orehovački, and Mario Đanić. Determination of Optimal Security Settings for LMS Moodle. Varaždin, Croatia: Faculty of Organization and Informatics, 2008. PDF.

[10] OWASP. "HttpOnly - OWASP." Main Page - OWASP. OWASP. Web. 16 Nov. 2011. <https://www.owasp.org/index.php/HttpOnly>.