SECURITY ISSUES IN OPEN ACCESS SOFTWARE MOODLE - A CASE STUDY

G. Mikadze, M. Khachidze, I. Lomidze, G. Tomadze

I. Javakhishvili Tbilisi State University
13 University Str., Tbilisi 0186, Georgia
grigol.mikadze976@ens.tsu.edu.ge, manana.khachidze@tsu.ge,
ilia.lomidze712@ens.tsu.edu.ge,
giorgi.tomadze784@ens.tsu.edu.ge

(Received 18.02.2024; accepted 18.06.2024)

Abstract

Bugs pose significant challenges in many software systems, especially in open-access software. Educational process management information systems are notably affected due to their heavy reliance on such software. This paper focuses on bugs in Moodle, one of the most common software systems, using data from a specific university. The study examines the extent of damage caused by these bugs and their impact on system functionality. Based on the results, the paper provides recommendations for identifing errors and the potential damage they may cause. The research findings can be useful for developers who decide to create or improve educational process management systems.

1 Introduction

Information and communication technologies have become an integral part of the modern world. With technological advancments, there has been a rise in the availablity of free-access programs across various fields, including education. Course management systems used in the field of education are distinguished by their diversity in functionality and popularity. Besides functionality, several critical factors influence the popularity of these systems, including the value, cost of the software, and the material, technical, and financial resources required for implementation, management, and updates.

There are many software programs that belong to learning platforms or course management systems (CMS) and have full and/or partial open access. Open access is one of the determining factors for the promotion of these systems. Among these types of systems, Moodle is one of the most common.

Moodle is the most widely used system in the field of higher education in Georgia, used by nearly all state universities and over half of private universities. The number of users increased significantly during the COVID-19 pandemic, as did the system's functional applications. Before the pandemic, Moodle was primarily used for distributing educational materials. However, during the pandemic, it became essential for various knowledge assessment procedures, including quizzes, exams, online surveys. Consequently, there was a notable increase in both registered and guest users, data volume and transaction numbers. As the number of Moodle users increases, so does the volume of their personal data. Accordingly, ensuring the security of both personal and non-personal data becomes a critical priority. Security deficiencies can lead to violations of personal information or the exploitation of users. These issues may arise from incorrect course planning, misconfigurations, or inherent flaws within the platform itself.

The aim of this research is to identify potential security breaches, assess the extent of system penetration, and determine effective methods for addressing security vulnerabilities, including unauthorized access.

The research was conducted using data from one of the major universities in Georgia. The data sources included both real and simulated data. During data extraction and processing, any data that could enable personal identification was completely excluded to ensure privacy.

2 Security-related bugs and their impact on software

Testing is one of the crucial steps in the software development and implementation process, as it helps identify various bugs. Equally important is the process of bug fixing, which occurs during the operation of the software.

The most common classification of bugs includes the following: **Boundary bugs** (1), which cause unexpected system behavior when data input falls outside the permissible range of values; **Performance bugs** (2), when the software does not perform optimally under conditions of heavy user load or large volumes of data; **Compatibility bugs** (3), caused by the software working differently on different platforms, browsers, and devices; **Usability bugs** (4), which affect user experience due to poor system design and complex processes; **Security bugs** (5), which threaten the integrity of the system or the protection of confidential information; and **Regression bugs** (6), where updated code inadvertently reverts issues that were previously resolved.

The wide variety of software types contributes to the diversity of bugs encountered. Since our research focuses on an open-access educational pro-

cess management system, we will specifically address the bugs that impact the security of this type of software. Learning management systems manage sensitive data, including student grades, personal information, and organization's intellectual property. According to data from Statista (7), educational organizations experienced an average of 2,314 cyber attacks per week in 2022. This is a 44% increase from the weekly average of 1,605 attacks in 2021. Learning portals face several security issues that typically arise from specific types of action vulnerabilities: initial contact, verification, segmentation, and integrity assurance. As previously mentioned, LMS platforms store sensitive information that can be the target of cyber attacks. Consequently, data tampering (alteration) and unauthorized access pose significant threats to these systems.

To mitigate these risks, it is essential to ensure comprehensive security functionality. Authentication is regarded as one of the most important functions (8), with the reliability of this process heavily influencing the main security threats. Typically, the authentication process starts during the initial contact with the system, followed by verification. Segmentation and breach of integrity are commonly targeted in hacking attacks.Among the most common types of authentication—password-based, multi-factor (MFA), certificate-based, biometric, and token- based—passwords are the most common authentication method. However, MFA (9) and single sign-on (SSO) (10), (11) are highly recommended. MFA may involve various forms such as temporary codes, secondary devices, or biometric authentication. The integration of multiple layers of security significantly complicates system penetration for hackers.

Another challenge is phishing, where attackers send fraudulent emails or text messages with malicious links to users. Typically, these links lead to websites designed to steal user account credentials or to distribute system-damaging malware. The primary source of such messages is often known as "spam".

Several recommendations have been proposed for the Moodle system to protect users from issues arising from spam (12).

Outdated systems are susceptible to vulnerabilities, as a lack of regular updates diminishes their resistance to threats. When technical specialists develop a security system, hackers often seek alternative methods of attack. Therefore, it is crucial to consistently update software to the latest security versions to maintain robust protection (13), (14).

In addition to technical challenges, adherence to data protection regulations is important. Non- compliance with regulations such as the General Data Protection Regulation (GDPR) (Consulting, 2016) can have severe consequences. GDPR stipulates how a company should respond to a detected attack, including identifying and containing compromised systems,

notifying the relevant supervisory authority and affected individuals, and patching vulnerabilities. Failure to adhere to these regulations can lead to violations and legal liabilities (15).

3 Bugs in Moodle

The empirical study, which analyzed real-world examples and security incidents, aimed to investigate the validity of expected threats posed by vulnerabilities in Moodle's security framework. Data penetration emerged as the primary concern, potentially caused by various factors.

Moodle is open-access software that requires continuous monitoring. To support this process, there are established general guidelines for information systems, one of which includes the use of TrackIt system. For instance, the Moodle Tracker system enables the identification and resolution of bugs within Moodle.

The choice of research method was largely influenced by the analysis of previously identified flaws in Moodle. In the real environment (where the data for the experiment was sourced), the results of these errors were documented and accessed using the Tracker system.

Different versions of Moodle contain various types of bugs, thereby escalating the risks of data breaches and illicit activities by malicious entities. Searchsploit, which has a public archive of software bugs, can provide information about known bugs. Specifically, by consulting Searchsploit Moodle (16), we can obtain detailed information about the vulnerabilities present in different Moodle versions.

Our study focuses on analyzing bugs and their harmful effects associated with Moodle LMS version 4.0, using data obtained from its usage. Our current analysis reveals that this version is susceptible to Cross-Site Scripting (XSS) issues-a type of injection where malicious scripts are injected into other websites (17). Consequently, attackers can exploit this vulnerability to

impersonate other users, steal session cookies, disrupt website functionality, or spread viruses (18).

4 Research methods

As noted, one of the challenges with Moodle is the XSS issue (19), a type of security vulnerability prevalent in some web applications. A site scripting vulnerability could be used by attackers to bypass access controls. Generally, there are two main types of XSS flaws: non- persistent (reflected) and persistent (stored). Non-persistent occurs when data provided by a

web client is immediately used by server-side scripts to analyze and display a results page to the user without adequately cleaning the content. Persistent, on the other hand, involves data provided by the attacker being stored on the server and subsequently displayed on regular pages to other users during their browsing sessions (20).

To test and identify issues, studies were conducted using real data from LMS used by a university that uses Moodle's open-access software. This system provides the sharing of lecture materials, various student evaluation procedures, and communication between teachers and students. The ultimate objective of the study is to assess the platform's vulnerability and pinpoint the causes related to the XSS issue.

The system has 835 users: 35 are teachers (with partially administrative rights) and 800 are students. Additionally, there are 62 educational courses available.

The initial challenge involves gathering information about defects. Upon overcoming this stage, several questions emerged, which were addressed based on the analysis of the research findings. These questions include:

1. What potential damage could these defects cause?
2. To what extent was it possible to penetrate the system and escalate the damage?
3. What is the underlying cause of the error?
4. What considerations should be made when configuring the Moodle system?

From the initial analysis, facilitated by the Tracker system, it became evident that Moodle uses weak cookies, which could potentially allow session hijacking.

We needed to "steal" the cookie. Specifically, when entering values into the Moodle search field, malicious code was inserted directly into the page's code without modification. In "inspect element", it was observed that the written words fit in one of the tags of HTML, in quotation marks, necessitating an escape. For instance, entering $< script > alert(1); < /script >$ would result in the code being enclosed in quotes. A successful XSS attack was executed using the following code:

$$" >< imgsrc = \#onerror = alert(document.cookie) >$$

The outcome confirmed that Moodle indeed had an XSS vulnerability, which was demonstrated by the retrieval of a session cookie displayed as an alert.

The stolen cookies enabled us to authenticate without a password. For the experiment, we used Cookie Quick Manager to insert the stolen cookie into the browser. After refreshing the page, we accessed the "victim user" page without needing a password. This event is attributed to an XSS vulnerability. Such breaches can lead to various errors, which have been explored experimentally.

The analysis revealed that an attacker could potentially steal the victim's session cookie, allowing them to hijack the session and access the victim's user page without a password. Other types of attacks, such as phishing, are also feasible. An attacker could create a fraudulent login page. If there is no validation on the input field, the attacker can directly inject malicious code.

Even if there is validation on the input field, the attacker could place their code elsewhere and execute it from that location. Consequently, if a potential victim is deceived into entering their passwords, the attacker could capture those credentials.

Note: This is not a case of stored XSS, meaning the attacker's malicious code is not transferred and saved in the database. However, the visibility of keywords in the GET parameter poses a significant risk. The user may believe they are clicking on a typical Moodle link, but in reality, this action executes the attacker's malicious code.

The next phase of the experiment focused on identifying vulnerabilities resulting from the configuration. The analysis revealed that unencrypted traffic (HTTP) was being exchanged between the client and the server, creating an opportunity for a Man-In-The-Middle (MITM) attack. This type of cyberattack involves an attacker secretly intercepting and relaying messages between two parties who believe they are communicating directly with each other (21). This part of the study aimed to assess the damage caused by the attack.

For this attack, we enabled IP forwarding in the Linux system by writing '1' in the file '/proc/sys/net/ipv4/ip_forward' (which is set to '0' by default). This modification allowed us to redirect traffic between the victim's computer and ours.

To initiate the attack, we used the arpspoof program. The victim's computer was targeted using the '-t' option, followed by the IP address of the person (i.e., the router), whose identity we were spoofing. We then used Wireshark to filter the incoming traffic from the victim's computer.

Consequently, when the "victim"; accessed the HTTP site (in this study, Moodle), we were able to capture their username and password.

This allowed us to log into the user's account from the victim's computer.

The results from filtering the traffic revealed that the data sent via the

POST method, including the password, was accessible (see Figure 1).

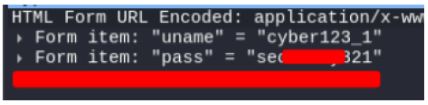

Figure 1: Sent data.

Additionally, a vulnerability associated with HTTP is that during a MITM attack, an attacker could modify the victim's data passing through their network card, such as injecting JavaScript (as demonstrated in our case).

In the experiment, the Kali Linux tools bettercap and beef-xss were used.

An 'http.proxy.injectjs' and 'arp.spoof' attack was conducted on the network using bettercap. Consequently, the attacker's computer attempted to inject the beef-xss script into all users engaged in unencrypted communication.

When accessing Moodle from a different computer, the attacker's computer successfully altered the data by embedding its own script, which then gained full access to the victim's browser and various other details.

Figure 2 displays the beef-xss panel, which provides detailed information about the victim's computer.

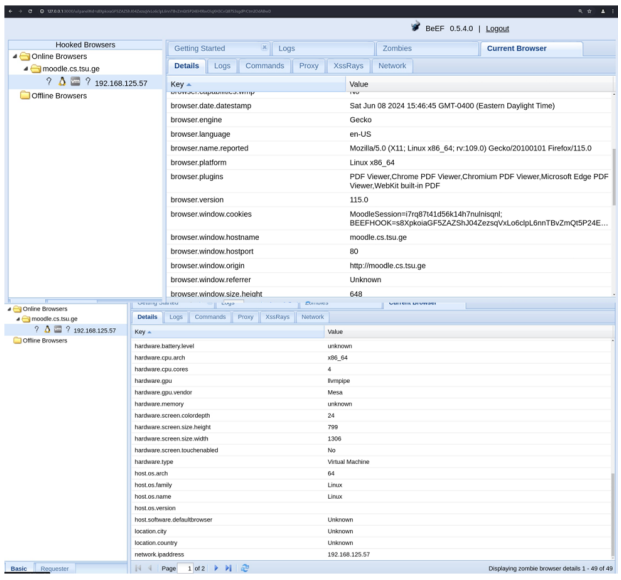

Figure 2: Victim's computer data.

beef-xss possesses one of the most dangerous features: it can steal every cookie from the victim's browser.

By pressing a button, a prompt could be displayed in the victim's browser, requesting them to log into Moodle. This prompt would persist until the required fields were completed.

Any data entered into these fields from the victim's computer was then transmitted to the beef- xss panel on the attacker's computer.

These two attacks can be executed in real-time during the online assessment process of students. In the first scenario, the hacker could view the answers submitted by other students to the tests. Furthermore, if the test is reviewed and correct answers are provided to the students at the end, the attacker could also access the correct answers as provided by the lecturer during the review.

In the second scenario, the attacker can alter the answers submitted by the victim, thereby tampering with the student's submission. The attacker could also falsify student results, steal exam papers, or gather personal information during the exam. Moreover, storing exam questions prepared by the lecturer in advance poses a risk, as it could lead to unauthorized access to the academic staff's user pages.

Research in this area has shown the critical importance of proper configuration (18). We have observed the severe consequences of software misconfiguration. This issue is managed locally. The experiment revealed that the primary configuration problem is the unencrypted data exchange (HTTP) between the client and the server. We recommend switching the protocol from HTTP to HTTPS and enabling HTTP Strict Transport Security (HSTS), a straightforward and widely supported standard that ensures browsers always connect to the website via HTTPS (22). By implementing these measures, it will be possible to prevent MITM and SSL Strip attacks (downgrading from HTTPS to HTTP (23)), significantly reducing the risk of information tampering.

5 Summary of Results

Consequently, Moodle, similar to other open-access software, contains several bugs that jeopardize data security and the educational process. Our research on the platform identified two types of issues: those caused because of incorrect configuration and those due to bugs in the software version.

The vulnerabilities discussed above enable an attacker to steal sessions, personal data, and login credentials from the victim, resulting in both data theft and disruption of normal system operations.

Although it is impossible to achieve complete immunity from attacks,

it is essential to establish barriers that enhance defense mechanisms. The empirical study presented in this paper enabled us to analyze the problem using real data and at least partially outline a path toward a solution. This approach will contribute to ensuring data security for universities using the Moodle platform.

Despite continuous updates since its release on August 20, 2002, and its main module being open access, the development of management systems like Moodle remains relevant. The findings from our research will be valuable for developers involved in this field, providing insights on how to identify bugs and the potential damage that can occur if these issues are not resolved.

References

1. Guo, X., Okamura, H., Dohi, T. Optimal test case generation for boundary value analysis. *Software Quality Journal*, (2024).

2. Han, X., Yu, T. An empirical study on performance bugs for highly configurable software systems. *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, (2016)

3. "Bug-for-bug compatible". (n.d.). catb.org. Retrieved from `https://catb.org`.

4. Grinberga, S. Software usability: Concepts, attributes, and associated health problems. *Proceedings of the Latvian Academy of Sciences, Section B Natural Exact and Applied Sciences*, **70**, 5 (2016).

5. Alhazmi, O. H., Woo, S.-W., Malaiya, Y. K. Security vulnerability categories in major software systems. *Proceedings of the Third IASTED International Conference on Communication, Network, and Information Security* (Jan 2006).

6. Braz, L., Fregnan, E., Arora, V., Bacchelli, A. An exploratory study on regression vulnerabilities. *Accepted at ESEM'22*, September 2022, Helsinki, Finland.

7. Average weekly number of malware attacks in organizations worldwide in 2022, by industry. Statista (February 2023). Retrieved from `https://www.statista.com`.

8. Papathanasaki, M., Maglaras, L., Ayres, N. (n.d.). Modern authentication methods: A comprehensive survey. *Computer Science and Robotics Technology*, 1-24 (2022). `http://dx.doi.org/10.5772/acrt.08`.

9. Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., Koucheryavy, Y. Multi-factor authentication: A survey. *Cryptography*, 1-31 (2018). https://doi.org/10.3390/cryptography2010001.

10. Shaikh, N., Kasat, K., Jadhav, S. Secured authentication by single sign-on (SSO): A big picture. In: *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 951-955 (2022). Greater Noida: Sharda University. https://ieeexplore.ieee.org/xpl/conhome/10037246/proceeding.

11. Laskaris, J. Security and the LMS: Your various options with eFront-Pro. April 26 (2016).

12. Moodle community. Reducing spam in Moodle. (2018). Retrieved from https://docs.moodle.org/404/en/Reducing_spam_in_Moodle.

13. Opigno.org. Data protection and privacy in LMS: What makes a solution secure? May 2 (2023).

14. Bruce, R. LMS security – What you need to know. Accord LMS. October 8. (2020). Retrieved from https://accordlms.com.

15. I. Consalting. General Data Protection Regulation. June 6. (2016). Retrieved from https://gdpr-info.eu/.

16. Teacher Hackthebox. (2019). Retrieved from https://executeatwill.com/2019/05/01/Teacher-hackthebox/.

17. Shao, W., Gao, Y., Song, F., Chen, S., Fan, L., He, J. A comprehensive empirical study of bugs in open-source federated learning frameworks. October (2024).

18. Kirsten, S., Manico, J., Williams, J., Wichers, D., Weidman, A., Jex, A., Smith, A., Knutson, J., Imifos, E., Yalon, R., kingthorin, V., Khanna, V. (n.d.). Cross-site scripting (XSS). Retrieved from https://owasp.org/www-community/attacks/xss/.

19. Alenazi, S. Moodle LMS 4.0 - Cross-site scripting (XSS) (2022). Retrieved from https://www.exploit-db.com/exploits/51115.

20. Web Application Security Consortium. (2005). Cross-site scripting.

21. Yasar, K., Cobb, M. What is a man-in-the-middle (MitM) attack? April (2022). Retrieved from https://www.techtarget.com/iotagenda/definition/man-in-the-middle-attack-MitM.

22. HTTP Strict Transport Security. Retrieved from `https://https.cio.gov/hsts/`.

23. Yackel, R. What are SSL stripping attacks? August 19 (2021). Retrieved from `https://www.keyfactor.com/blog/what-are-ssl-stripping-attacks/`.