Applying Security to Moodle Grades

S. Oraá Pérez[1], **C. Hernández Díez**[2], **and J.A. Marcos García**[2]
[1]Research Group: Technologies Applied to Innovation, Isabel I University, Burgos, Spain
[2]University of Valladolid, Valladolid, Spain / Isabel I University, Burgos, Spain

Abstract— *Moodle is an open source course management system that helps universities and high schools to provide online courses for their students. However, Moodle is insecure and grades could be easily changed. In this paper, we propose a solution to detect any change in Moodle grades, illegal or not, and notify the users in charge to keep grades secure.*

Keywords: Moodle grades, Information security, Database security

1. Introduction

Changes in labour market and the demand for qualified staff, as well as in ongoing training, have led many people to balance work and studies. The development of technology has also helped to the integration of online education not only in universities but also in high schools. This increase in demand and technological advances have led to the appearance of online universities that allow students to obtain a degree while working. In addition, new tools, such as Moodle, have been developed just for that purpose [1].

Moodle is an Open Source Course Management System (CMS) under the GNU General Public License, also known as Learning Management System (LMS) or Virtual Learning Environment (VLE) [2]. Being an open-source project, Moodle is continually being reviewed and improved on to suit users' current and evolving needs. It can be customised in any way and tailored to individual needs. Moodle is also developed in several languages, enabling people in many countries to use it without any language limitation.

Moodle is a learning platform designed to provide educators, administrators and learners with a single robust, secure and integrated system to create personalised learning environments. For all those reasons, the use of Moodle has been extended up to 80,896 registered sites with 12,028,828 courses [3]. Figures 1 and 2 show the distribution by country and the countries with the highest number of registered sites.

Since Moodle is a widespread tool, it is more likely to be attacked and some sensitive information could be altered. The security implemented in Moodle will not prevent some kind of attacks, such as the ones directed to the grades stored. That is why our objective will be to protect those grades encrypting them and detecting any change.

The rest of the paper is organized as follows. Section 2 gives some relevant information about Moodle. Section 3 discusses security vulnerabilities in Moodle. Section 4 presents the solution we have implemented to improve security. Section 5 shows the set of tests that the system has undergone for evaluation. Section 6 presents future work in this field to improve our solution. Finally, section 7 concludes the paper.

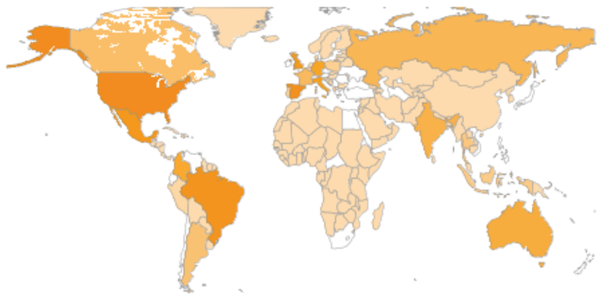
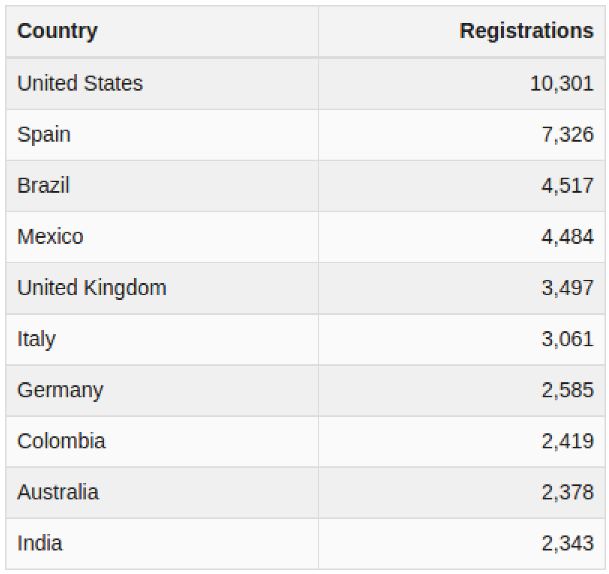


Fig. 1: Moodle Known Sites by Country



Country	Registrations
United States	10,301
Spain	7,326
Brazil	4,517
Mexico	4,484
United Kingdom	3,497
Italy	3,061
Germany	2,585
Colombia	2,419
Australia	2,378
India	2,343

Fig. 2: Top 10 from registered sites in 234 countries

2. Moodle in a nutshell

As we have stated, Moodle is a particularly widespread tool in the education environment. It has features making it possible to be used by hundreds of thousands of students at the same time, in both university and primary education. Many institutions use it as a platform for online education while others use it as support for classroom education; this method is called blended learning.

Moodle has had several versions (Figure 3), however, 2.X and 3.X are the most extended ones. We have developed our system using the database tables available in these versions.

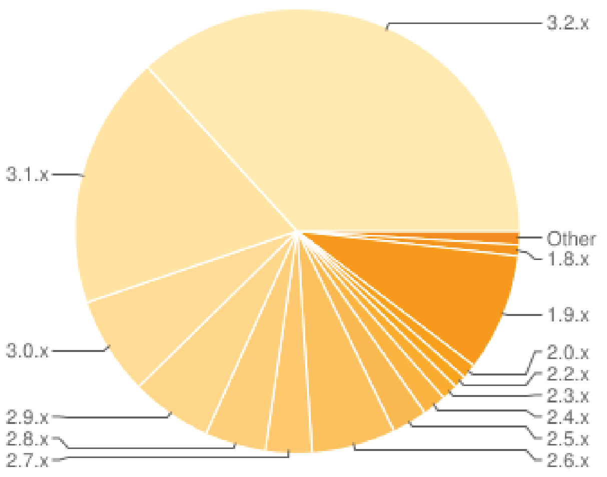


Fig. 3: All Moodle registrations by version

Moodle is divided into sections called contexts where roles can be assigned. These contexts can be: the whole system, collections of courses (categories), courses, blocks or activities. We are interested in every context that can have a grade associated such as activities or courses.

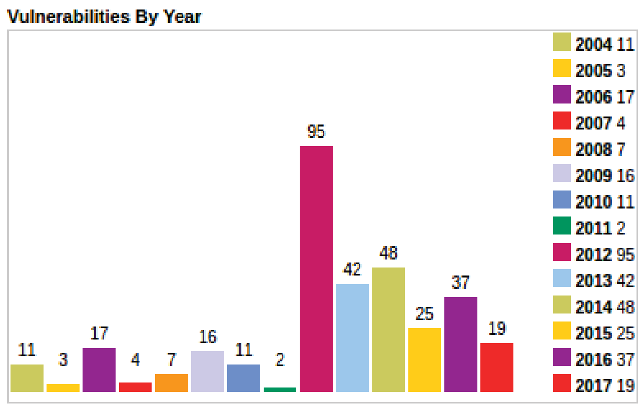
Likewise, Moodle has eight roles by default; however, only 4 of them can change grades using the graphical interface: "Managers" associated with the context "system" that can modify all the grades in it. "Course Creators" which are enrolled with the role "Professor" once the course is created. "Teachers" and "Non-Editing Teachers" that are associated with the context "course" and can modify all the grades in it. And finally "System Administrators" that have permission to do anything. However, it should be noted that "System Administrators" are not considered a normal role in 2.X or 3.X Moodle versions, therefore, to prevent them to be treated as intruders, this kind of accounts must be also associated with the role "Manager".

Finally, there are several ways in Moodle to indicate the completion of a course such as activity completion, completion of other courses, date, enrolment duration, unenrolment, achieve a score, manual completion by the user and manual completion by others. We will use the date criterion to determine the day in which the grades will be considered as final. To turn on the completion criteria, one of the Administrators or Managers in Moodle must enable the tracking completion of courses and then assign an end date to the courses.

3. Security vulnerabilities in Moodle

Moodle, as any other collaborative tool, implements some security. The LMS can selectively limit and control access of the diverse community of users using roles. Online content, available resources and some of the functions provided by the tool may be limited to privileged users.

Nevertheless, Moodle is an insecure platform. In fact, some Web Pages exist exclusively dedicated to detect insecurities in this tool [4] as well as a whole subforum in the official page [5]. In Figure 4, we can see the vulnerabilities occurred in the past years and the more common types [6]. Among all these attacks, we would like to highlight SQL injections, which are a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution to, for example, dump the database contents to the attacker or alter it.



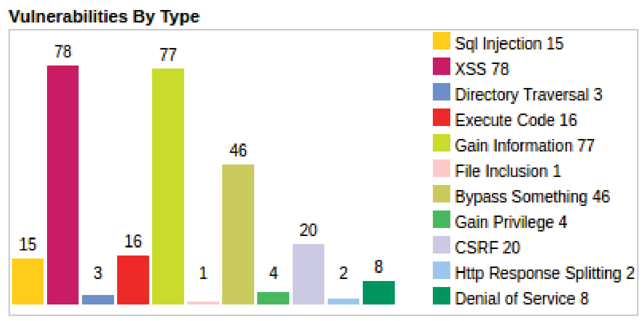


Fig. 4: Vulnerabilities by year and type in Moodle

All of this proves that the security implemented in Moodle to protect grades and other types of information, does not prevent most kind of attacks. Using Moodle as the main and, on some occasions the only tool in online education, requires improving existing security measures. For example, Moodle grades are stored as numbers in the database using the gradebook. Once the grade is stored, if it is changed using the graphic interface, Moodle keeps a historic of the changes but if someone have stolen the teacher password or the grade is changed using a SQL injection, Moodle will not be able to detect it or keep a record of the change.

Grades are sensitive information and in official degrees it is required by law to keep them unaltered and secure for years. We considered that the most frequent actions against the integrity of the grades will be the ones that alter the value stored in the database. That is why our solution was designed to detect any change in those values, whether the Moodle graphic interface or a SQL injection is used, and to add extra protection using encryption. However, it will still be the responsibility of the assigned professor to detect the addition or deletion of activities associated to grades.

4. A solution

Php [7] is the language used in Moodle development. That is why it was selected as the language in our system, so any other Moodle developer will be able to understand or improve it.

4.1 How it works

The system allows making security copies of Moodle grades improving security. It also detects illegal attempts to access and allows people in charge to fight them.

Our system requests the information from Moodle database and retrieves it automatically every 15 minutes, reducing timespan potentially available for intruders to make any modification. When the received grades are not in the system, they are stored in our database system in two ways, explicit to speed up searches and encrypted to enhance security. When the grades are already stored in the system, the system proceeds to do a comparison to detect if there has been a change in them. If an inconsistency in the grades is detected, the system will send a notice to the user which amended that grade the last time, according to Moodle records, informing about the situation and asking for a confirmation of the change.

Once an inconsistency in the grades is detected, the system will not check again that grade until the relevant user resolves the conflict, confirming to the system the real value of the grade. At a certain time of the day, if notifications have not been attended, the user will get a reminder. After selecting the correct value, the system will automatically update the existing grade.

If the user that changed the grade does not have the privileges to do it, the system will identify him as an intruder and alert the System Administrator. It will be the Administrator's responsibility to check the grades and maintain the consistency in both databases and, if necessary, to notify the teacher in charge.

Nevertheless, our system does not change the values stored in the Moodle database to avoid inconsistencies in the related tables.

Our main goals are, first of all, to securely store the grades of the activities and courses so teachers do not have to check the grades to be sure there was not an illegal change in them. Second, to notify the teacher, manager or relevant user in case of any alteration in the grades, illegal or not, is detected and keep a log of the changes. And third, to securely store the grades for some time in case of future claims, as requested by law.

4.2 Data selection

The connection to the Moodle database will only be established for 3 reasons: every 15 minutes to get the information related to grades; to get the data of "Managers", "Teachers" and "Non-editing Teachers"; and to obtain the data of the intruder to inform the "System Administrator", in case the user that modifies the information does not have the privileges to do it.

Before we design the sql query to retrieve the information, we must consider the user that might be receiving the information: the numeric IDs of courses, activities and students will not have any meaning to the teachers. For that reason, it would be necessary to know the name of those items when an inconsistency is detected, so the teacher will be able to check the change using the Moodle graphic interface. However, IDs will be useful for the System Administrator.

It would also be necessary to obtain the teachers full name for reference, and a transformation of the grade modification date since Moodle stores dates in Timestamp format.

All the required information is requested in a single query as can be seen in Query 1.

Query 1: Grades

```
SELECT
  item.id,
  item.itemname,
  item.courseid,
  grade.userid,
  user.firstname,
  user.lastname,
  grade.finalgrade,
  grade.timemodified,
  item.itemtype,
  grade.usermodified
FROM
  moodle.mdl_grade_items item,
  moodle.mdl_grade_grades grade,
  moodle.mdl_user user
WHERE
  item.id = grade.itemid and
  user.id = grade.userid and
  courseid not in (
    SELECT
       course
    FROM
       moodle.mdl_course_completion_criteria CC
    WHERE
       (CC.criteriatype = 2 and
    UNIX_TIMESTAMP
    (DATE_SUB( sysdate () ,
      INTERVAL 30 DAY)
    )>= CC. timeend ));
```

The Moodle tables used to obtain this information are:

- "Grade_items" contains all the activities that can have an associated grade. Our query only returns already graded activities. The fields with relevant data are:
  - "Id": Unique identifier of the activity in the database.
  - "CourseId": Course ID which the activity belongs to.
  - "ItemName": Name of the activity.
  - "ItemType": Type of the activity. Relevant when you want to obtain the global course grades. In that case the value must be "course".
- "Grade_grades" contains the numerical grades. The fields with relevant data are:
  - "ItemId": Identifier of the activity which the grade belongs to.
  - "UserId": Student ID that owns the grade.
  - "UserModified": ID of the user who modified the grade the last time.
  - "FinalGrade": Numerical value of the grade.
  - "TimeModified": Date of the last modification.
- "User", contains information about the users of Moodle. Fields with relevant data are:
  - "Id": Unique identifier of the user in the database.
  - "FirstName": Name of the user.
  - "LastName": Last name of the user.

Finally, to optimize the algorithm, we searched a way to prevent further requests of the grades in a course after 30 days since its completion. After this time, we consider that the grades will not be modified and the grade encrypted in our database will be final. The Moodle table used to obtain this information is:

- "Course_completion_criteria" which contains the completion criteria associated to a course. The fields with relevant data are:
  - "Course": Identifier of the course.
  - "CriteriaType": type of the completion criterion. Value 2 represents a date.
  - "TimeEnd": Contains a date, if criteriatype = 2.

The obtained information is stored in the following tables (Figure 5). The table "Grade_Security" relates the grades with the users in a concrete activity and course. To reduce the allowed time to make any modification, Query 1 is executed every 15 minutes, to avoid any unnecessary join operation between tables we store all the information in the same table. Our system also keeps a log of the detected changes.

Some other queries are necessary to obtain the relevant information of privileged users (teachers, managers...) or intruders [8].

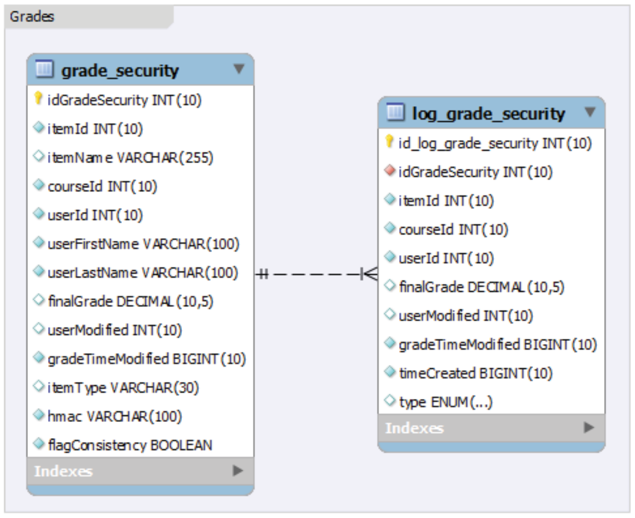


Fig. 5: Grade Tables

4.3 Information Security

Our system has been designed to increase grade security in Moodle. To that end, we do not only need to make a backup of the grades in plain text but regularly check that the stored information has not been altered. If these measures were the only ones implemented, our system would remain vulnerable to the same kind of attacks as the Moodle database, for example SQL injections. That is why, in the security database associated to our system, we store the same information but encrypted [9].

To encrypt the data our system uses the HMAC function, i.e., Hash-based message authentication code. HMAC is a specific construction to calculate a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authentication of a message. Any cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC. We use SHA-2 that produces a 256-bit hash value (Figure 6) [10]. The mathematic function that defines HMAC is:

$$HMAC(K,m) = H((K \oplus opad)||H((K \oplus ipad)||m))$$

- H(): hash function.
- K: private key. If its size is smaller than a block size, zeros will be added on the right.
- m: raw text, before the encryption.
- ||: concact.
- $\oplus$: XOR operator.
- opad: exterior padding (constant "5c" repeated B times).
- ipad: interior padding (constant "36" repeated B times).
- B: block size.

To illustrate the behaviour of this function we provide a scenario. We have a course with three students and a teacher. They have two gradable activities, a Moodle quiz and the

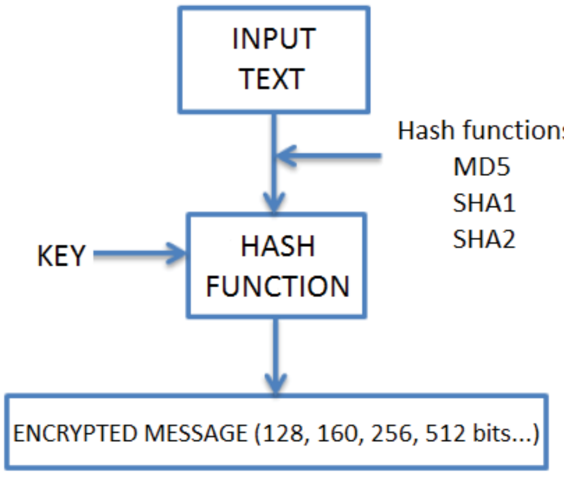


Fig. 6: HMAC encryption

final grade of the course. Once our application detect the grades stored in Moodle, it would request the necessary data and store them in our system database. The initial values available in Moodle are shown in Table 1.

Table 1: Moodle Information

Data	Student 1	Student 2	Student 3
CourseId	754	754	754
UserId	**1**	**2**	**3**
FirstName	Javier	Patricia	Laura
LastName	Gil	Carballude	Pascual
ActivityId	**475**	**475**	**475**
ItemName	Exam	Exam	Exam
FinalGrade	7.5	3	6.2
TimeModified	1494001839	1494002139	1494002019
ItemType	Quiz	Quiz	Quiz
UserModified	89	89	89
ActivityId	**489**	**489**	**489**
ItemName	Final Grade	Final Grade	Final Grade
FinalGrade	8	3	6
TimeModified	1495815819	1495816119	1495816239
ItemType	Course	Course	Course
UserModified	89	89	89

Before grades are introduced in the database in our application for the first time, it will be verified that they have been introduced by an authorised user (userModified). In our case, the user ID "89" is assigned to Alfredo Cabria, the teacher for this course. If no malicious access is detected, the system will use the key chosen by the administrator to encrypt grades and to store them along with the remaining information obtained.

Using the test key "test1", and encryption SHA-256, we obtain the values included in Table 2. Those values will be stored in our database. The flag for grade modifications will remain at "0" until a modification is detected. Every 15 minutes, the system will retrieve the grades in Moodle and will compare them with encrypted values, adding any new grade, if necessary. In the next section, we will explain the scenario we created, and how our application will work in case of inconsistencies or modifications by non-authorised users are detected.

Table 2: Initial Grade Values

Grade	SHA-256
7.5	58564d89db89d2a60715e72edffcf23a67630736dd08 78dc4377a3088c426560
3	dce833faaf325bd4a741becff5e2c6428c616acc3590d 6e6c37353f24519fb6f
6.2	04c6d25e76e0e44f88f6d2fc87890b3397c3361a3385b 9f9be112da31ab104b8
8	fa0e5c59f0529214dcb362c916a1102f5d130736742d 70ca6522b41f0069e5af
3	dce833faaf325bd4a741becff5e2c6428c616acc3590d 6e6c37353f24519fb6f
6	5e3f43dc9c7249b1c9d485681420372b68ff0f5f8b916 f2892501e9b66d4f18a

4.4 Notice Generation

As we have stated in the previous section, when a grade that previously exists in our system database is detected, the system applies a HMAC encryption algorithm and the result is compared to the encrypted information stored. If a conflict between the stored encrypted information and grades obtained appears, a second test will be performed, which will detect whether the integrity of our database security has been compromised.

Let's see how it works through our example. In Table 3 we can see that one of the two grades awarded to "Student 1" (Javier Gil) remains the same (7.5), whereas the other grade has been modified from 8 to 9. Grades are compared one after the other, following the same order as they have been awarded, so the 7.5 grade will be encrypted and verified against the stored value first (Table 3). As no discrepancy is detected, our application will move on to compare the remaining grades. The next one (9) will undergo the same process of encryption and verification against the stored value. As soon as the system detects that both values are different, a security test in the database is run. To do so, the system will retrieve the value stored in the database (8), which will be encrypted to obtain the correspondent value. This value, in turn, will be compared with the stored encrypted value in the table in order to check if it has been compromised. As both values continue, the system will continue.

When after this second check, the system detects that our database security has not been compromised, the system will generate a notice that informs the relevant user about the detected inconsistency. In our example, an e-mail will be sent to teacher Cabria to inform him about a modification detected in Javier Gil's grades. The teacher will have to confirm the change or report a malicious modification. When the relevant user resolves the conflict, the system recalculates the encrypted information stored and updates the database information if necessary. In our example, the

Table 3: Grades After 15 Minutes

Moodle	SHA-256	Our App	SHA-256
7.5	58564d89db89d2 a60715e72edffcf2 3a67630736dd08 78dc4377a3088c4 26560	7.5	58564d89db89d2 a60715e72edffcf2 3a67630736dd08 78dc4377a3088c4 26560
9	acf03bc1c919ed0 14f7b3f3f8079ec5 b49e47b5e77983 a20fa86727c1dec 689a	8	fa0e5c59f052921 4dcb362c916a11 02f5d130736742 d70ca6522b41f0 069e5af

teacher confirms the modification as made by himself just by clicking on the final grade, which is sent to our system. The grade will be recalculated to obtain an encrypted grade (Table 3), which will in turn be stored in the database. In addition, a new entry in the log table keeping track of changes will be created.

If the modification had not been made by teacher Cabria, measures would have been implemented to ensure that the grade shows the real value. The teacher's password and access privilege will be updated to prevent future unauthorised changes.

Administrators, rather than teachers or course supervisors, will be notified in two cases. First, if grades have been modified by unauthorised users (users without a role allowed to do such changes). That is, if the field userModified linked to the grade is connected, for instance, to a student. Second, if an inconsistency between the value stored in the encrypted field HMAC value and the information stored in clear is detected. In our example, if a different HMAC value from the stored one is obtained, it will be treated as a serious security breach. This will allow to detect any SQL injections on the security database. The System Administrator will be notified of this situation every 15 minutes. The notifications could be stopped and the grade would not be checked again until the inconsistency is solved.

5. Evaluation

To evaluate this system we followed the validation and verification criteria of Ian Sommerville [11].

We used point to point testing before the full implementation of the system. We have conducted periodic inspections of the different modules of the application with the objective of detecting defects in the implementation of the system. We have corrected certain code errors that could prevent the correct system operation such as connection failures to the databases, or produce an unstable state, such as failed insertions or misleading information in the database. To ensure that the whole system was checked we used several tests.

Use-case tests check individually every use-case to ensure that the functionality is correctly covered. To do this, we tested the sequence associated to every operation and, in addition, every query in Moodle and our system database. Likewise, we performed flow analysis, checking the inputs and outputs of all functions to ensure they deliver the desired outputs. This also served to detect erroneous calls. The most significant test undertaken were: obtain the grades, connect to the database, detect changes in the grades, discover illegal accesses and changes, find legal changes, make log insertions, search notifications without answer to solve the inconsistency, notify an illegal access to the administrator of the system, solve the inconsistency and avoid any other change when the inconsistency was already solved.

Trajectory tests cover all possible scenarios of a use-case and each and every one of the bifurcations in each decision making them separately, ensuring that everything runs at least once. We conducted these tests paying special attention to the system functions that make database queries and combinations of functions that share parameters.

Black Box Testing detects if the output is correct for all inputs proven regardless of the code. Given our system is modular, we tested all the functions of the system making sure that the system was able to detect any changes, inconsistencies in the grades and database security failures. The queries were also tested using inputs with correct and complete data and inputs with insufficient data, such as grades without any activity associated and activities without a grade.

Finally, we used White Box Testing to detect any failure in the structure or implementation of the code. To do so, we tested every variable ensuring that all were used, well initialized, non redundant and to make sure their names do not cause conflicts with other variables of the same application.

Table 4 shows some examples of tests run [8].

Table 4: Some examples of the tests performed

Item	Examples
Grades	Obtaining new values. Value changes detection. No grades. Finished course. Log insertions.
Users	Obtaining the authorized users list. Modification make by authorized and unauthorized users.
Illegal Accesses	Changes on the application data.
Database Connections	Moodle Database. Application Database.
Notice Generation	Changes detected. Relevant user (Administrator or Teacher).
Conflict resolution	Authorized users. Unauthorized users. Administrator: changes on the application data.

Stress tests that would overcharge the system until it stops running, as well as Statistical testing that is designed to reflect the frequency of user inputs and to make a reliability estimation of the system, could not be performed on local environments. Non meaningful results were obtained to check the system in the real environment, as it did not have enough grades, activities or users. These tests are currently being run in an actual environment, as explained in the section “Discussion & Future Work”.

6. Discussion & Future Work

The system stores the activities and course grades in a secure way without the intervention of the teacher; notifies the teacher in case a change in the grades is detected, whether authorized or not, revealing the illegal alterations and keeping a log to obtain statistics; stores the encrypted grades for future claims as provided by law; and detects illegal modifications in our system database. The queries implemented work with every Moodle 2.X and 3.X versions.

However, our system will only notify the teacher and keep the record of the changes, it will be the teacher’s responsibility to keep the Moodle database updated. In a future line of work the system will be able to update this database without the intervention of the teacher.

The system notifies the teachers using emails. This method could be changed, for example, to use a ticket system.

We are currently working on stress tests with 5000 users, 600 courses and about 100 grades for each student. We expect that these tests allow us to improve the performance and reliability of our system.

7. Conclusions

The need for a tool that allows online universities to provide their students with an online classroom has led to the development of tools such as Moodle. However, this kind of tools are not secure enough and important information can be manipulated and altered without the consent or the knowledge of the teachers in charge. That is why some alternative ways to protect data must be implemented.

Our system is capable of controlling the stored grades in Moodle 2.X and 3.X for any degree and course, to avoid unnoticed illegal modification of the grades using the graphic interface or directly in the Moodle database. Furthermore, the security has been improved using the HMAC encryptation (SHA-256 with key) assuring the security of every grade.

Acknowledgments

The authors would like to thank Isabel I University for giving them the opportunity of working on this idea for an online University and testing it in a real environment.

References

[1] S. Uusiautti, S. Määttä, E. Leskisenoja, Succeeding Alone and Together-University Students’ Perceptions of Caring Online Teaching, *Journal of Studies in Education*, No.7, 2017.

[2] *Moodle.* (undated). [Online]. Viewed 2017 May 08. Available: `https://moodle.org/`

[3] *Moodle Statictis.* (undated). [Online]. Viewed 2017 May 08. Available: `https://moodle.net/stats/`

[4] *CVW Details. The ultimate security vulnerability datasource.* (undated). [Online] Viewed 2017 May 08. Available: `http://www.cvedetails.com/vulnerability-list/vendor_id-2105/product_id-3590/Moodle-Moodle.html`

[5] *Moodle. Security and privacy.* (undated).[Online] Viewed 2017 May 08. Avaible: `http://moodle.org/mod/forum/view.php?id=7301`

[6] *CVW Details. Security and privacy.* (undated). [Online] Viewed 2017 May 08. Avaible: `http://www.cvedetails.com/product/3590/Moodle-Moodle.html?vendor_id=2107`

[7] *PHP: Hypertext Preprocessor.*(undated). [Online]. Viewed 2017 May 08. Available: `http://php.net/`

[8] S. Oraá, *Aplicando Seguridad en las Calificaciones de Moodle,* 2012.

[9] T. R. Peltier, *Information Security Policies, Procedures, and Standards: guidelines for effective information security management*, CRC Press., 2016.

[10] D. L. Evans, P. J. Bond, A. L. Bement. (2002). *The Keyed-Hash Message Authentication Code (HMAC).* Viewed 2017 May 08. Available: `http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf`

[11] I. Sommerville, *Software Engineering*, 7th ed. Addison-Wesley, 2005.