www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(4), 2026, 252-268. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i4.1318

Development of a Blockchain for voting: A security approach in the student process at UAEH

Braulio J. Sanchez Urbina[1], Evangelina Lezama León[1], Alonso Ernesto Solís Galindo[1], Israel Acuña Galván[1], Sandra Luz Hernández Mendoza[1]

[1] Universidad Autónoma del Estado de Hidalgo, Carretera Tizayuca-Pachuca Km. 2.5, 43800 Fuentes de Tizayuca Hidalgo.
✉Corresponding author: evangeli@uaeh.edu.mx

Abstract. This project develops an educational web platform that integrates an algorithm capable of detecting SQL vulnerabilities in PHP-based web page code. The project aims to strengthen the computer security knowledge of sixth-semester students in the Bachelor's Degree in Information Technology at the Tizayuca Higher School (UAEH) through the use of a practical tool that analyzes code, offering automated feedback in simplified, non-technical language. The research takes a mixed approach using an agile methodology based on DevOps (Development Operations), allowing iterative cycles of development, testing, and continuous improvement. The algorithm was validated with real code fragments, enabling correct detection of vulnerabilities in code in more than 75% of cases, reducing false positives by 10%. The results show a positive educational impact, significantly improving students' understanding of SQL injection prevention, thereby promoting the adoption of good security practices on websites.

Keywords: SQL Injection, Cyber Security, IT Security, Web Page, algorithm, ChatBot.

Article information
Received: Feb 01, 2026
Accepted: April 11, 2026

1 Introduction

Maintaining a secure website is essential for organisations that rely on digital systems to provide products or services. Web security involves protecting websites, applications, networks, and data against unauthorised access, manipulation, theft, and service disruption (Fortinet, n.d.). For students enrolled in the Information Technology programme at the Escuela Superior de Tizayuca, the growing demand for web-based services creates professional opportunities while also raising an important question: Are students adequately prepared to develop secure web applications?

Injection is included among the principal web-application security risks identified in the OWASP Top 10. SQL injection vulnerabilities may occur when an application incorporates unvalidated or unsanitised user input into dynamically constructed database queries. Successful exploitation may allow an attacker to access, modify, or delete information and may compromise the availability and reputation of the affected system (OWASP Foundation, 2021).

Online learning environments can support flexible and autonomous learning by allowing students to determine when and how long they engage with educational activities. They may also incorporate interactive resources and practical exercises that facilitate the application of knowledge (Alvarado, 2021).
Accordingly, an academic platform focused on web security could help students understand SQL injection vulnerabilities and learn how to prevent them. Integrating an automated code-analysis system into such a platform may further support learning by identifying insecure practices, explaining the associated risks, and suggesting appropriate corrections.

2 Background information

Researchers have developed several automated approaches for detecting and predicting SQL injection attacks. Palma Ponce (2022) evaluated a multilayer perceptron intended to identify SQL injection attacks in big-data environments. The study reported limitations in the model's ability to recognise certain textual patterns, which reduced its effectiveness in detecting all attack instances.

Fig. 1. *Log analysis interface implemented in the multilayer perceptron system (Spanish).*

Llumiquinga Guamba (2022) evaluated a support vector machine–based system for detecting and predicting SQL injection attacks in big-data environments. The results indicated limitations in classification accuracy associated with the characteristics and variability of SQL injection patterns. These findings suggest that preventing insecure code during development may be more appropriate in educational contexts than attempting to predict attacks after deployment.

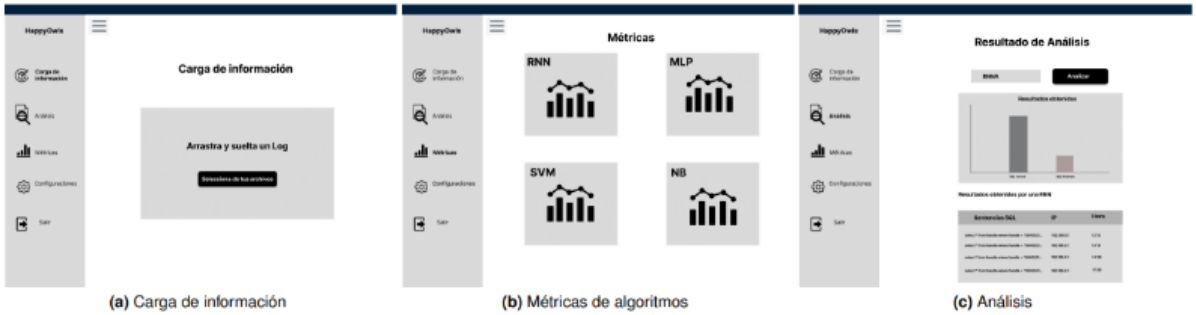

Fig. 2. *Interface modules for the project proposed by Llumiquinga Guamba (Spanish).*

A different approach is provided by Burp Suite Professional, a security-testing toolkit that combines manual and automated techniques for analysing web applications, HTTP requests, input parameters, application endpoints, and potential vulnerabilities (PortSwigger, n.d.-a). A PortSwigger case study describes Burp Suite Professional as an important component of the security-testing workflow used by Microsoft Azure security

engineers. The case study identifies Taylor O'Dell as the Microsoft security engineer who provided the testimony; therefore, the source must be cited as PortSwigger rather than as an independent publication by O'Dell (PortSwigger, n.d.-b).

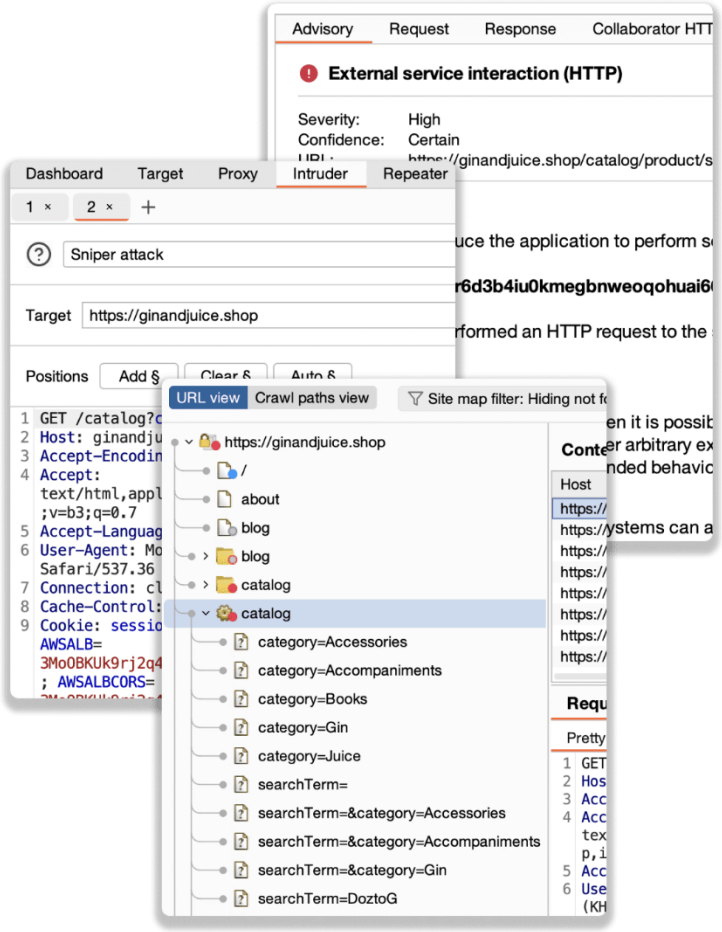

Fig. 3. *BurpSuite's main interface when analyzing a project*

One of the main drawbacks of Burp Suite is both the cost of its license (at least $500 USD, or MXN $10,000) and its complexity of use (primarily used by security experts such as penetration testers, security engineers, etc.), which limits its accessibility in educational settings or for students in the early stages of their training.
Upon analyzing these projects, we can see that the most feasible options for preventing security issues on websites can be summarized as follows:

- Using third-party software to analyze code
- Learning how to secure websites before entering the workforce

In order to better examine each of the features introduced by previous projects, a structured comparison was conducted between the various existing approaches to the detection and prevention of SQL injections. This comparison takes into account factors relevant to both professional and educational settings, including implementation cost, target audience, ease of use, detection capability, and applicability to educational environments.

Tool / Approach	Cost	Target audience	Complexity	Detection capability	Educational suitability
Multilayer Perceptron (Palma Ponce; M.B, 2022)	Low	Researchers	Average	Limited to complex patterns	Average
Sistema basado en SVM (Llumiquinga Guamba; A.M., 2022)	Low	Researchers	Average	Moderate accuracy	Average
Burp Suite (PortSwigger, N.D.)	High(~$500 USD)	Penetration testers, security engineers	High	High detection rate through dynamic scanning	Low Educational Suitability for Beginners

Fig. 4. *Feature Comparison Table by Project*

As can be seen in the comparison table above (Fig. 4), machine learning-based solutions have limitations when attempting to identify complex patterns associated with SQL injections, which reduces their practical effectiveness. On the other hand, professional tools, such as the one offered by PortSwigger, provide high detection capabilities through dynamic scanning techniques and HTTP request analysis, although their complexity and cost limit their accessibility for students or educational environments.

3 Proposition || Submission

The proposed solution is a detection system based on regular expressions, which analyzes a webpage's code to identify any vulnerabilities. It generates both the corrected code and an explanation of the changes made—including why they were made—using simple, non-technical language. This allows anyone who uses it (Fig. 5) to understand these vulnerabilities, know how to correct their errors, and, most importantly, avoid making these mistakes in the workplace.

Tool / Approach	Cost	Target audience	Complexity	Detection capability	Educational suitability
ExoSQL (propuesta)	None	IT students	Low	Focused on prevention and understanding vulnerabilities	High

Fig. 5. *Features table for the proposed solution, ExoSQL*

ExoSQL aims to fill a unique niche within the security tools ecosystem: providing an accessible and educational environment designed to teach SQL vulnerabilities as part of the academic training of students in the technology field.

To make it easier for any user to interact with the pattern-based detection system, we propose integrating it into a chatbot (Fig. 6) within an educational platform, allowing not only for learning focused on the use of a rule-based analyzer but also for supplementing that learning with explanations, sample code, and more.

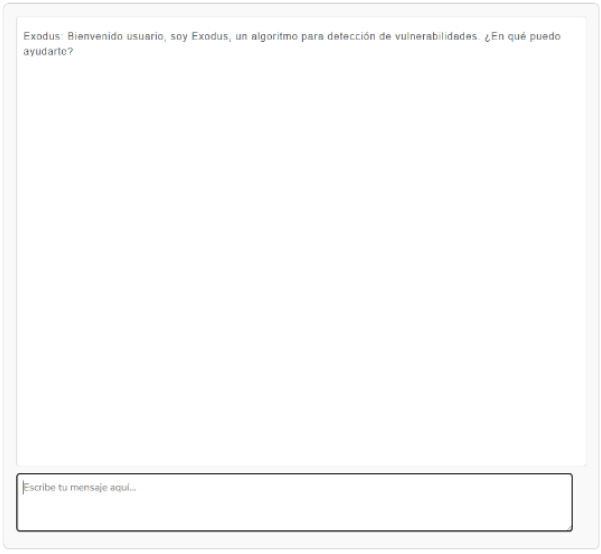

Fig. 6. *Proposed chatbot interface, based on similar chatbots*

The goal of the proposal is simple: to allow users to communicate with the chatbot and submit their code, enabling the algorithm to analyze the code, reconstruct it, and return it "fixed" (Fig. 7), along with a brief explanation of the proposed code changes, areas for improvement, and the reasons behind the changes.

Fig. 7. *Screenshot of the chatbot in action after analyzing vulnerable code*

The detection system is designed to analyze PHP code using text patterns; that is, it analyzes user input to identify the intent of the user's message, separating and analyzing the raw PHP code through a function that extracts SQL patterns, variables, and tags that match HTML and PHP syntax, thereby enabling the system to distinguish between user text and actual code.

First, the chatbot displays a welcome message that is pre-programmed into the code. After that, the user is prompted to enter text. When the user presses the ENTER key, the text input field is locked, and the user's text begins to be analyzed.

First, the user's intent is analyzed—that is, what they are trying to convey in their text. This intent is defined by four "requests":

- **req_analize:** Here, by using simple text patterns—such as detecting the use of words like "help," "analyze," "evaluate," "need," etc.—we can determine that the user is requesting a code analysis, so we proceed to the next step: Code reconstruction
- **req_greetings:** Here, the system detects that the user is "greeting" the chatbot. Therefore, if a **req_analize** request is not detected at this point (since it takes precedence over this type of request, as explained below), the chatbot will simply greet the user and ask if they would like to analyze some code
- **req_thank:** This detects whether the user wishes to express gratitude upon receiving the expected (or useful) result to their problem. This request was added with the possibility in mind that the user might want to express gratitude (something quite common when using chatbots, according to The New York Times, 2025).
- **req_unknown:** If a specific request cannot be detected, the system returns "unknown request" by default, meaning that the system has failed to identify the correct intent and will not proceed with code analysis (by default, this has a weight of 0)

Each request has a "weight" (i.e., a priority for when it is returned), so the higher the weight, the higher the priority, and therefore, it will be returned before the others:

- **req_unknown** has a weight of **0**, allowing it to be overridden by other requests
- Both **req_thank** and **req_greetings** have a weight of **1**, since they only interact with the user without changing the code
- **req_analize** has the highest weight, which is **2**, because it ensures that, regardless of whether the user thanked or exited, if they later request "analyze" (i.e., another "analyze" request is detected), the analyzer will know to ignore the other requests (or set them aside) and reanalyze the code

After analyzing the request, the chatbot will notify the user that code analysis has begun and will start analyzing the user's input. To do this, the user's plain text is separated from the code using text patterns that detect any HTML or PHP tags, extracting the functional code line by line. To do this, follow these steps

- **Code sanitization:** First, the code is "cleaned up" by removing tags such as "<" and replacing them with something safer or removing them entirely.
- **Source mapping:** Here, all PHP sources in the code (GET, $_POST, REQUEST, etc.) are analyzed and sanitized
- **Capturing variable assignments:** Here, any assignments are captured, as well as inline calls in the code. Mysqli expressions are also captured here, as well as procedural MySQL expressions.
- **SQL statement detection:** Here, all variables that appear to contain SQL (SELECT, INSERT, etc.) are detected, as well as variables used in inline calls
- **Generation of prepared statements:** Finally, prepared queries are generated for every candidate statement; that is, the code is rewritten using regular expressions to replace unsafe assignments with prepared statements, removing assignments enclosed in quotes, removing spaces, ensuring that variables are always used by reference, while at the same time preserving the logic and functionality of the original code
- **Code reconstruction:** In this section, the modified code is assembled, cleaned up again so that the chatbot can write it, and returned to the chatbot as a response. Here, the chatbot will load the analysis completion message and attach

To make this learning experience more impactful, it was proposed that the pattern-based detection system be integrated into an educational website, which would contain additional information about SQL injections, how they work, and how to prevent them. To this end, a website was developed using HTML5 technology, with CSS and the Bootstrap framework for design and responsiveness, JavaScript for functionality, and the AJAX framework to connect JavaScript with PHP, allowing the pages to retain their HTML nature without being limited by PHP's functionalities.

The website will be divided into the following sections:

1. **Home:** This section provides information about the project, its relevance to the job market, and links to the registration and login pages (Fig. 8)

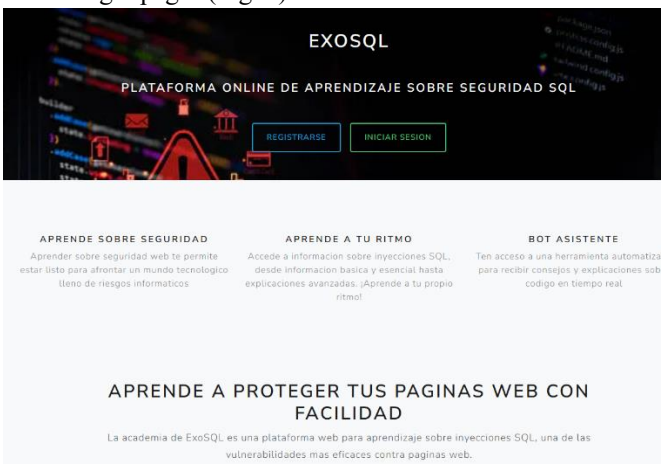

Fig. 8. *ExoSQL homepage, along with information about the project (Spanish).*

2. **Registration:** This section displays a registration form where users must enter their personal information to access the system (Fig. 9).

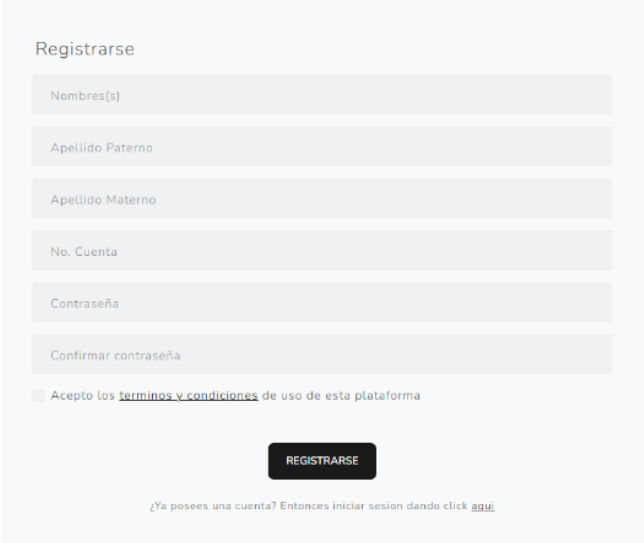

Fig. 9. *Registration form for logging into ExoSQL, using typical registration data*

Since the project is aimed at students at a Mexican High School, an email verification process was implemented (Fig. 10), whereby a 6-digit verification code (unique and non-repeatable) is generated upon registration and sent to the user's institutional email address (Fig. 11)

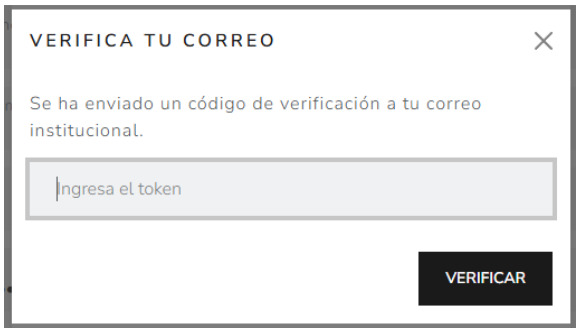

Fig. 10. *Identity verification method using a unique token sent to the user's email address (Spanish).*

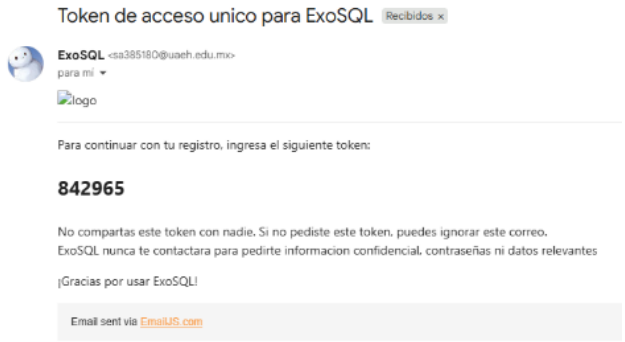

Fig. 11. *Automated email sent via EmailJS, along with the validation token (Spanish).*

The email address is generated based on the information provided in the registration form, thereby preventing users from using any email address to register

3. **Login module:** This section features a simple form that allows users to access the page by entering their account number and password (Fig. 12).

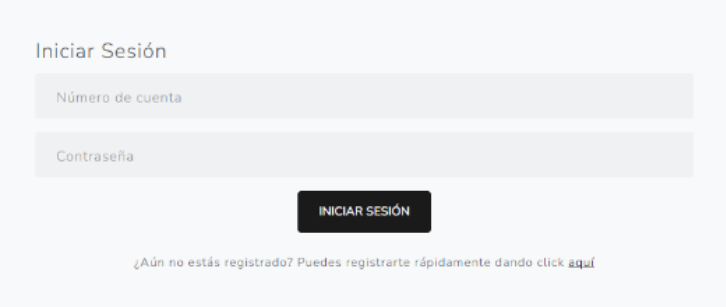

Fig. 12. *Login form, with password hashing (Spanish).*

4. **User Welcome Module:** This section provides a brief introduction to the user and includes two buttons for navigating the page (Fig. 13)

Fig. 13. *Home page, after logging in to ExoSQL (Spanish).*

5. **Web Academy Module:** Here, a series of cards are displayed that cover various topics related to web security, with a particular focus on SQL injections (Fig. 14). Here, users can click "Learn More" to open a modal with additional information on the related topic, as well as (if applicable) a code snippet or example of how it works (Fig. 15).

Fig. 14. *Interface for the safety tips module, based on a simple touchscreen system (Spanish).*

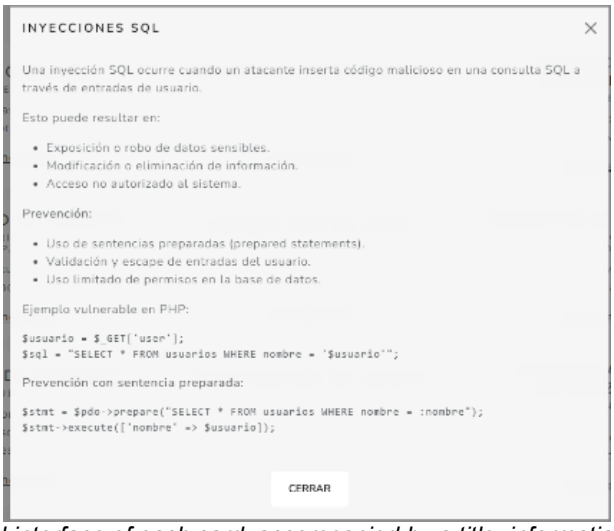

Fig. 15. *Internal interface of each card, accompanied by a title, information, and practical examples with code (Spanish).*

In addition to these sections, two additional sections were created: "Terms and Conditions" (Fig. 16) and "Privacy Policy" (Fig. 17), which address various topics of interest related to terms of use, legal matters, data usage, information privacy, and so on.

Fig. 16. *User interface for terms and conditions, featuring expandable accordions (Spanish).*

Fig. 17. *User privacy policy interface with expandable accordions (Spanish).*

The proposed solution not only helps users access up-to-date information on SQL injections in web pages, but also supports their learning by using non-technical language that is easy to understand and explained in simple terms, thereby avoiding the frustration that often comes with trying to learn new topics that are typically complex—and, in many cases, presented in a language they do not fully grasp due to its complexity.

The development of this system provides a solution to a problem that has existed since 1998, contributing to the education of IT students in a non-intrusive way, giving them access to constantly updated information in a language they do master and using simple language. This not only provides broader access to this type of information but also helps reduce this cybersecurity problem by addressing the root cause.

4 Results

Evaluation of the pattern-based detection system

To evaluate the performance of the rule-based analyzer, a set of 40 PHP code snippets was used, structured to represent different scenarios in which code might be written by users and how it interacts with a database, analyzing the various interactions with the databases. The snippets were designed to include both vulnerable and secure code, with the aim of thoroughly analyzing the system's ability not only to detect vulnerable code but also to avoid modifying code that does not require updating.

In addition, code snippets with various programming structures and unique implementation styles were incorporated to assess the system's performance with real users.

The vulnerable code snippets included various common patterns of SQL injection vulnerabilities, such as:

- Dynamic query construction through the concatenation of form variables
- Use of variables without validation
- Direct insertion of parameters into SQL queries
- SQL queries with simple variables containing command injection points

In addition, various code snippets were incorporated that implement secure practices, such as prepared statements and input validation, with the aim of evaluating the system's behavior when compared to correctly implemented code.

The tests performed on the system focused on determining:

- The accuracy of the analysis: whether it correctly detected intended vulnerabilities.
- The frequency of **false positives** (code flagged as vulnerable when it is not).
- The frequency of **false negatives** (vulnerable code that went undetected).
- The algorithm's response speed (average analysis time per code).

Initial results showed that the analyzer was able to detect more than 75% of the vulnerable cases (Fig. 18), with a false positive rate of less than 10%.

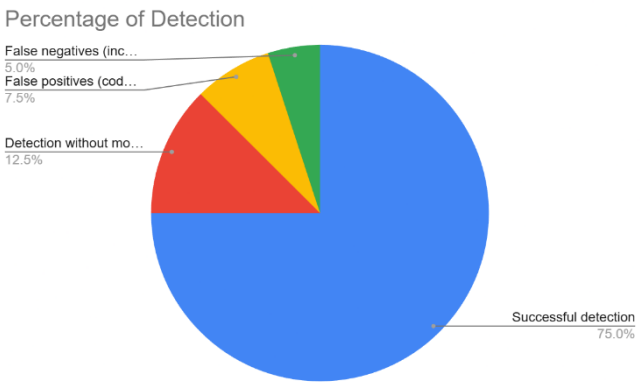

Fig. 18. *Results chart showing the algorithm's detection rates, with a success rate of 75%*

It is important to note that the dataset size was manually designed to include both vulnerable and secure cases, enabling an accurate analysis of the system's ability to detect SQL-based vulnerability patterns, as well as allowing for the estimation of false positives and false negatives in a controlled environment.

Educational Impact Assessment of the Platform

Following successful testing of the algorithm, a controlled study was conducted with students in a web development program, in which the group of students was divided into two groups:

- **Control group:** This group continued with its traditional education without any changes
- **Experimental group:** In addition to their standard education, this group would use the platform with the pattern-based detection system

To assess the impact of using the platform, diagnostic tests were administered before (pre-test) and after (post-test) the period of platform use. To facilitate comparison between the measurements, the difference (△) between the results obtained was calculated.

Indicator evaluated	*Control Group (pre-test)*	*Control Group (post-test)*	*△ Control*	*Experimental Group (pre-test)*	*Experimental Group (post-test)*	*△ Experimental*
Percentage of vulnerabilities correctly identified	**26%**	**30%**	**4%**	**32%**	**75%**	**43%**
Percentage of PHP code that follows best practices	**63%**	**65%**	**2%**	**60%**	**85%**	**25%**
Overall average in theoretical knowledge	**5.4**	**6**	**0.6**	**6.2**	**8.7**	**2.5**

Fig. 19. *Results of the study conducted among computer science students*

As can be seen in the table above (Fig. 19), the most notable change is observed in the identification of vulnerabilities, where the experimental group showed a **43%** increase compared to their initial assessment.

Similarly, the use of best practices in PHP code increased by **25%** compared to their knowledge prior to using the platform.

Finally, in order to quantify the magnitude of the change observed in the experimental group, the **relative improvement index** was calculated (Fig. 20). This index shows the percentage increase in the final result relative to the initial result, thereby quantifying the group's improvement from its starting point and reinforcing the comparison of the experimental group's academic performance against that of the control group (Fig. 21).

$$RI\ (\%)\ = \frac{Post\ -\ Pre}{Pre}\ x\ 100$$

Fig. 20. *Relative improvement index formula, indicating the percentage by which a result increased from its initial value*

Indicator evaluated	*Control Group (pre-test)*	*Control Group (post-test)*	*Relative Improvement Index (Control)*	*Experimental Group (pre-test)*	*Experimental Group (post-test)*	*Relative Improvement Index (Experimental)*
Percentage of vulnerabilities correctly identified	**26%**	**30%**	**15%**	**32%**	**75%**	**134%**
Percentage of PHP code that follows best practices	**63%**	**65%**	**3%**	**60%**	**85%**	**42%**
Overall average in theoretical knowledge	**5.4**	**6**	**11%**	**6.2**	**8.7**	**40%**

Fig. 21. *Tabla de resultados con índice de mejora relativa entre ambos grupos*

Based on this indicator, it can be observed that the ability to identify vulnerabilities increased by 134% for users who used the algorithm, while there was only a 15% increase for the control group; meanwhile, regarding the use of best practices in PHP, the experimental group improved by 42%, while the control group showed only a 3% improvement. Likewise, theoretical knowledge in the experimental group showed a relative improvement of 40.3% compared to 11% in the control group. These results reinforce the trend observed in the absolute values and suggest a positive impact of using the platform and the vulnerable pattern recognition system on the learning of concepts related to web application security.

Finally, a questionnaire was conducted to analyze the user experience and the quality of the platform, both in terms of educational aspects and user satisfaction.

The first questionnaire (based on user experience) focused mainly on analyzing the experience of using the platform, through an analysis of its design, colorimetry, and intuitive navigation. The results shown (Fig. 22) indicate that more than 50% of the users surveyed were completely satisfied with the website, and most of them had a positive opinion of it.

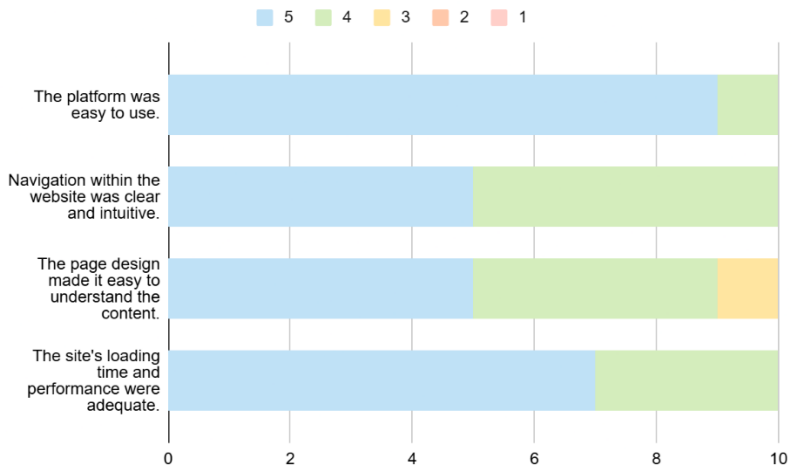

Fig. 22: *Results of the "User Experience" form*

The second questionnaire focused mainly on analyzing the platform's content, i.e., the information provided by the website, the algorithm's feedback when returning the user's code analysis, as well as the user's general opinion of the platform and whether, in their opinion, they consider that they have seen an improvement in website security issues.

The results (Fig. 23) show that more than 80% of users are satisfied with the content of the information on the platform, and that 80% of users consider that they have gained useful knowledge for computer security, thus confirming that the platform not only contributed to the user, but that the user themselves feels satisfied and ready to face new challenges in their career.

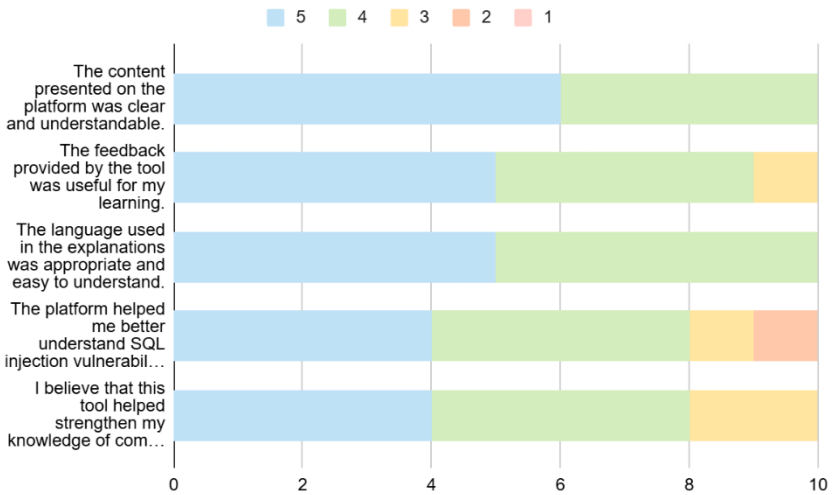

Fig. 23: *Results from the "Content Quality and Usefulness" form*

The third question aims to analyze the educational and practical aspects of the platform by asking users for their opinions and observing whether the platform not only contributed to their knowledge but also sparked their interest in continuing to learn about the topic, further enriching their understanding and further reducing the possibility of SQL vulnerabilities occurring on websites developed by them.

The results of this questionnaire (Fig. 24) show that 90% of users are curious to learn more about this topic, and 80% of them would definitely recommend the platform to others, in order to spark curiosity about website

security and learn about this important topic, thus getting more users to use the platform and also to seek information for themselves, thereby achieving the ultimate goal: to reduce the possibility that a user could make a mistake when developing a website and cause an SQL injection in projects developed by them.

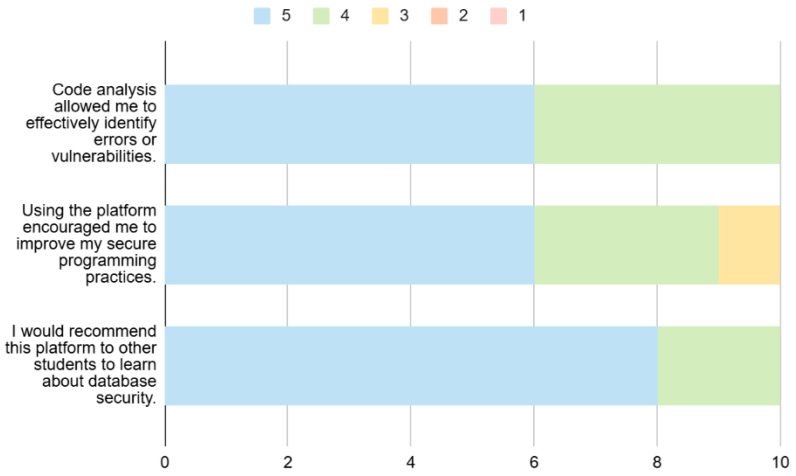

Fig. 24: *Results of the "Educational and Practical Considerations" Form*

Finally, the fourth questionnaire seeks to analyze overall user satisfaction, with the aim of finding out not only whether users consider that the platform meets their expectations, but also whether they wish to continue learning and whether, in the future, they would like to continue using this type of tool and platform to continue learning about similar topics.

The results obtained (Fig. 25) show a general acceptance of the project, revealing that at least 40% of users will definitely use similar tools to continue learning about these types of topics, which, although somewhat complex, can be learned easily and simply with the right tools.

In addition, it can be seen that the majority of users are satisfied with the use of the platform and consider that it has met their expectations and the goals set.

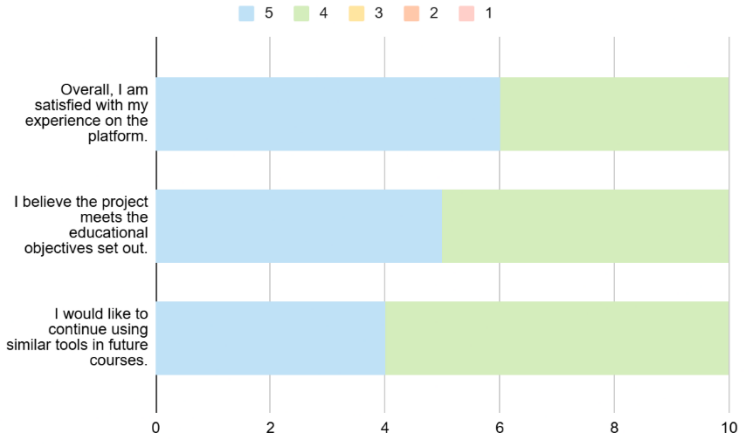

Fig. 25: *Results of the "Overall Satisfaction" Survey*

5. Discussion

Interpretation of results

The results obtained during the evaluation of the pattern-based detection system show that the developed system is capable of identifying a significant proportion of vulnerabilities related to SQL injection in PHP code snippets. The detection of more than 75% of vulnerable cases suggests that the text pattern recognition approach effectively identifies common forms of insecure SQL query construction.

Furthermore, a false positive rate of 10% indicates that the system is sufficiently accurate in distinguishing between vulnerable code and code that follows secure coding practices. This is significant because a high false positive rate reduces a tool's usefulness by generating unnecessary alerts for programmers.

Finally, these results suggest that the proposed algorithm is useful as a tool for the early identification of PHP web code, particularly in educational settings.

Educational impact

The results of the study conducted among technology students indicate a positive impact regarding the use of the proposed platform. The observed improvement in both the ability to identify vulnerabilities and the application of best practices suggests that integrating the scanner into an educational setting promotes the learning of web security concepts.

In particular, both the increase in the identification of vulnerabilities and the improvement in the use of best practices observed in the experimental group suggest that the use of the platform not only helps identify security flaws in web applications but also fosters the development of safer programming habits.

The relative improvement analysis reinforces this trend, as it reveals significant differences between the experimental group and the control group, suggesting that the use of automated support tools is a useful teaching resource for strengthening the instruction of security in web development

Study limitations

Despite the results obtained, it is important to consider certain limitations when conducting the study.

The system was evaluated using 40 manually constructed code snippets, which allowed us to analyze the algorithm's behavior in a controlled environment. However, we acknowledge that the size of the dataset limits the statistical significance of the results obtained when compared to large-scale code repositories.

Furthermore, the proposed system relies on text pattern recognition using regular expressions, which makes it possible to identify typical structures associated with SQL injection vulnerabilities. However, this approach may have limitations when dealing with complex scenarios found in more professional environments.

For example, regular expressions face challenges when identifying obfuscated queries or those constructed using variable transformations, thereby reducing detection capabilities in certain cases.

Furthermore, the system was developed specifically for websites written in PHP (since it is the most widely used language in web development education), so its direct application to projects using other languages (or frameworks) will require adapting the analysis patterns used.

Future work will focus on expanding the evaluation dataset by incorporating larger-scale code repositories, as well as code obtained from public vulnerability databases, thereby strengthening the initial results.

These limitations suggest that, while the system is useful as a tool to aid in vulnerability detection and as an educational resource, its professional use may require additional techniques capable of analyzing more complex code structures.

6. Conclusions

The results obtained throughout the implementation, validation, and evaluation of the Exo SQL system show that the system meets its primary objective: to facilitate the detection of SQL injection vulnerabilities in PHP code and to reinforce students' learning about secure development practices. These results allow the system to be viewed not only as an analysis tool but also as an educational resource for training web developers with a deeper understanding of web security and the use of secure practices.

The significant improvement in performance in the experimental group compared to the control group demonstrates that the platform is not only functional but also effective as a teaching tool. This result suggests that the use of automatic detection systems could complement traditional teaching approaches, facilitating learning by providing immediate feedback on vulnerabilities in the code developed by students.

The project also demonstrated that the use of regular expressions and pattern analysis was sufficient to cover the most common cases in an academic setting, although it is recognized that a more professional environment will require a shift toward more complex analyses.

Users report being satisfied with the results, while some mention being surprised that they were unaware of certain topics and how disconcerting it is to "learn about this until they enter the workforce." Others, however, expressed their interest by pointing out areas for improvement, such as:

- The system could include more detailed feedback for cases where no vulnerabilities are detected.
- The platform could be enhanced with visual examples or short explanatory videos.

The final conclusion is clear: the development and implementation of Exo SQL represents a functional, accessible, and educational solution for addressing the problem of SQL injections in academic projects. We successfully integrated an interactive experience with security fundamentals, and the results obtained validate its positive impact on the teaching-learning process.

References

Alvarado, M. (2021, November 26). *¿Qué es un curso en línea y cuáles son sus ventajas?* Luca. https://lucaedu.com/blog/que-es-un-curso-en-linea-y-cuales-son-sus-ventajas/

Deb, S. (2025, April 25). Decir "gracias" y "por favor" a ChatGPT cuesta dinero, pero quizás valga la pena. *The New York Times*. https://www.nytimes.com/es/2025/04/25/espanol/negocios/chatgpt-gracias-por-favor.html

Fortinet. (n.d.). *Seguridad web y seguridad de sitios web*. Retrieved March 14, 2023, from https://www.fortinet.com/lat/resources/cyberglossary/what-is-web-security

Llumiquinga Guamba, A. M. (2022). *Evaluación de algoritmos de minería de datos para detección y predicción de ataques de inyección SQL en big data: Evaluación de SVM para la detección y predicción de ataques de inyección SQL en big data* [Undergraduate thesis, Escuela Politécnica Nacional]. Repositorio Digital Institucional de la Escuela Politécnica Nacional. https://bibdigital.epn.edu.ec/handle/15000/23405

OWASP Foundation. (2021). *Introducción*. OWASP Top 10. https://owasp.org/Top10/2021/es/

Palma Ponce, B. A. (2022). *Evaluación de algoritmos de minería de datos para detección y predicción de ataques de inyección SQL en big data: Evaluación de un perceptrón multicapa para la detección y predicción de ataques de inyección SQL en big data* [Undergraduate thesis, Escuela Politécnica Nacional]. Repositorio Digital Institucional de la Escuela Politécnica Nacional. https://bibdigital.epn.edu.ec/handle/15000/23400

PortSwigger. (n.d.). *Burp Suite Professional is the cornerstone of Microsoft's pentesting toolkit*. Retrieved June 12, 2024, from https://portswigger.net/customer-stories/microsoft-case-study

PortSwigger. (n.d.). *Burp Suite Professional*. Retrieved March 16, 2023, from https://portswigger.net/burp/pro