nected remote to the antenna and receive data from RFID tag. Antenna and RFID Tag connected automatically and transmit data.

This study focuses on the feasibility, accuracy and security of RFID technology in user identification through empirical analysis and simulation experiments of RFID systems in different application scenarios. The results of the study show that RFID has significant advantages in terms of identification speed, accuracy and anti-interference.

Based on the above problems, the study proposes several recommendations to enhance the security of RFID user identification, including: using encryption to protect data, setting access control to restrict permissions, and adopting interference suppression strategies to enhance system robustness. In the future, with the advancement of data protection technology, the application of RFID technology in user identification is expected to be further popularised.

References

1. Apparatus and a method for testing radio-frequency tags : pat.US 2024/0369607-A1/Jesse Tuominen, Helsinki. – Publ. date 07.11.2024.

2. Tech Target Network. – Access mode: https://www. techtarget.com/iotagenda/–definition/RFID-radio-frequency-identification. – Access date: 07.11.2024.

УДК 004.056.5

STRATEGIES FOR ENHANCING DATABASE SECURITY AGAINST SQL INJECTION

Deng Yuanyuan
Belarusian State University of Informatics and Radioelectronics
e-mail: 0908dyy@gmail.com

Summary. *This article explores the concept, working principles, and potential harms of SQL injections while focusing on a variety of effective methods to prevent such attacks. These strategies enhance system protection against SQL injection and improve overall security.*

In the digital era, databases serve as the backbone of information storage and management, providing essential support for various applications and services across numerous industries. As organizations increasingly rely on data-driven decision-making and digital solutions, the security of these databases becomes critically important. One prevalent threat is SQL injection, a highly dangerous attack method used by cyber attackers to exploit vulnerabilities in web applications and compromise databases. These vulnerabilities typically arise when input fields and variables are not properly sanitized or validated, leaving the system exposed to malicious inputs. By leveraging these weaknesses, attackers can bypass security protocols and gain unauthorized access to sensitive data, allowing them to retrieve, alter, or delete user information. Such attacks pose a substantial threat to data integrity and confidentiality, underscoring the urgent

need for robust filtering and validation mechanisms to protect databases from unauthorized manipulation. To strengthen database security against SQL injection attacks, several key strategies can be implemented [1–2].

Use prepared statements. Prepared statements are a technique that processes SQL statements and parameters separately. This method compiles SQL statements before sending queries and uses placeholders to replace user input, thereby preventing the injection of malicious code. In this way, the database will only replace the placeholders with the parameters provided by the user when executing and will not treat them as SQL code. Python:cursor.execute("SELECT * FROM users WHERE email = ?", (userInput,))

Through the above examples, we can see how prepared statements are implemented in different programming languages, which can effectively prevent SQL injection.

Input validation and filtering. Whitelist strategy: only allow input that meets specific formats or conditions. For example, email addresses, phone numbers, etc. can all be validated by regular expressions to ensure that the input conforms to the expected format. Blacklist strategy: Block known malicious input patterns, but this method is not always reliable because attackers may use different encoding methods to bypass the blacklist. Character filtering: Remove or escape special characters such as ' , " ; etc. to prevent them from being interpreted as part of the SQL syntax. Data type validation: Ensure that the input data type is consistent with the expected, for example, integer fields only accept numbers.

The principle of least privilege; emphasize the permission control of database users. The principle of least privilege means that database users should only be granted the minimum permissions required to perform their work. For example, the database user of an application should not have permission to execute DDL (data definition language) statements, such as creating, modifying, or deleting tables. Create specific users: Create dedicated database users for each application or service to limit their access rights. Role management: Use the role management function of the database to define the permissions of different roles and assign users to corresponding roles.

Use ORM frameworks; ORM frameworks allow developers to interact with the database using object models instead of writing SQL code directly. This abstraction layer can reduce the risk of SQL injection because ORMs usually automatically handle parameter binding and input filtering.

Entity Framework (.NET): Provides LINQ queries to avoid direct SQL. Django ORM (Python): Safely interacts with the database through model definitions and query sets. Hibernate (Java): Automatically handles SQL statement generation and parameter binding.

Regular security audits. Performing regular security audits of code and databases can detect potential security vulnerabilities in a timely manner. Audits should include checking the input validation logic of the code, how database queries are constructed, and the configuration of user permissions. Automated

security auditing tools can help teams efficiently identify potential SQL injection vulnerabilities. For example.

SQL MapAn open-source tool for detecting and exploiting SQL injection vulnerabilities; OWASP ZAP: a comprehensive tool for discovering security vulnerabilities, including detection of SQL injections.

This article examines the concept of SQL injection and its related risks, emphasizing the critical importance of database security. It provides a brief overview of several common and effective prevention strategies that can significantly improve system security while safeguarding the integrity and confidentiality of sensitive data.

References

1. Halfond W. G. J, Viegas J and Oso A. A Classification of SQL Injection Attacks and Countermeasures. In Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation (SCAM), 2006. – P. 13–24.

2. Santos J, Ferreira, P. Preventing SQL Injection Attacks in Web Applications. In Proceedings of the International Conference on Cyber Security and Protection of Digital Services, 2020. – P. 1–10.

УДК 612. 821.6

A SYSTEM THAT ENCOURAGES A CAR DRIVER TO DRIVE RESPONSIBLY AND SAFELY

Dubovsky V. A., Savchenko V. V., Drozd V. V.
The Joint Institute of Mechanical Engineering of the National Academy of Sciences of Belarus
e-mail: vdubovsky.email@gmail.com

***Summary**. The aim of this paper is to propose a concept of system that encourages a car driver to drive as responsibly and safely as possible. The proposed concept can be used as a framework to develop a research project aimed to improve road safety by improving the behavior of road users.*

It is known that every year in the world about 50 million people are injured and more than 1 million people die in road accidents [1]. Moreover, road accidents are associated with economic losses, medical cost and insurance expenses [2]. It is also known that the main cause of all road accidents in the world (about 93 %) is human factor [3]. All this suggests that the road safety is a serious issue and that the most effective way to solve it is to improve driver behavior.

The objective of this paper is to create a concept of system that encourages a car driver to drive as responsibly and safely as possible.

The proposed concept is illustrated in the diagram (figure 1).