INJECTION VULNERABILITIES: ANALYSIS OF MALICIOUS CODE ATTACKS IN MODERN INFORMATION SYSTEMS

Uzbekistan, Sharda University, Faculty of Engineering and Technology, 1st-year Master's student
Obidjonov Nursulton G'ulomjon o'g'li.

Annotatsiya *Ushbu maqolada Injection zaifliklari, ya'ni tizimga zararli kod yuborish orqali amalga oshiriladigan kiberhujumlar tahlil qilinadi. Injection hujumlari asosan noto'g'ri tekshirilgan foydalanuvchi kiritmalari orqali yuzaga keladi va bu holat ma'lumotlar bazasi, server yoki ilova xavfsizligiga jiddiy tahdid soladi. Maqolada SQL Injection, Command Injection va Cross-Site Scripting (XSS) kabi asosiy turlari, ularning kelib chiqish sabablari hamda oldini olish usullari yoritilgan.*

Kalit so'zlar: *Injection, SQL Injection, XSS, Command Injection, kiberxavfsizlik, OWASP, ma'lumotlar bazasi xavfsizligi*

Annotation: *This article analyzes injection vulnerabilities, which are cyberattacks performed by inserting malicious code into systems. Injection attacks occur mainly due to improper validation of user inputs and pose serious threats to databases, servers, and applications. The paper discusses major types such as SQL Injection, Command Injection, and Cross-Site Scripting (XSS), their root causes, and effective prevention techniques.*

Keywords: *Injection, SQL Injection, XSS, Command Injection, cybersecurity, OWASP, database security*

Аннотация: *В данной статье рассматриваются уязвимости типа Injection, то есть кибератаки, основанные на внедрении вредоносного кода в систему. Такие атаки возникают из-за неправильной обработки пользовательского ввода и представляют серьёзную угрозу для баз данных, серверов и приложений. В работе описаны основные виды*

атак, такие как SQL Injection, Command Injection и Cross-Site Scripting (XSS), их причины и методы защиты.

Ключевые слова: *Injection, SQL Injection, XSS, Command Injection, кибербезопасность, OWASP, безопасность баз данных*

INTRODUCTION

With the rapid growth of web applications and distributed systems, cybersecurity threats have become increasingly sophisticated. One of the most critical vulnerabilities affecting modern systems is Injection. Injection attacks occur when an application fails to properly validate or sanitize user input, allowing attackers to insert malicious code into a system. According to the OWASP Top 10 security risks, injection vulnerabilities have consistently been among the most dangerous threats to web application security. These vulnerabilities can lead to unauthorized data access, data manipulation, system compromise, and in severe cases, complete server takeover.

Injection attacks exploit the trust between an application and its input sources. When user input is directly included in database queries, operating system commands, or application scripts without proper validation, attackers can manipulate the system's logic and execute unintended operations.

RESEARCH METHODOLOGY

Technical Foundation of Injection Vulnerabilities. Injection vulnerabilities occur when an application processes untrusted input without proper validation, allowing attackers to interfere with the intended execution flow of interpreters such as SQL engines, operating system shells, or scripting engines. The core problem lies in the failure to separate data from commands, which enables malicious input to be interpreted as executable code.

Modern applications often rely on dynamic query generation and user-driven input processing. When this input is directly concatenated into queries or commands, the system becomes vulnerable to injection attacks. The

fundamental principle violated here is input trust boundary separation, which is a key concept in secure software design.

SQL Injection remains the most widely exploited form of injection vulnerability. It occurs when attackers manipulate SQL queries to gain unauthorized access to a database.

- Attackers can:
- Bypass authentication mechanisms
- Extract sensitive database records
- Modify or delete data
- Execute administrative database commands

SQL Injection typically arises in applications that construct queries using string concatenation instead of parameterized queries. For example, login forms that directly insert user input into SQL statements without validation are highly vulnerable.

Advanced SQL Injection techniques include: Blind SQL Injection (data extraction without visible output) Union-based SQL Injection (combining malicious queries with legitimate ones) Time-based SQL Injection (using delays to infer data) .The impact of SQL Injection can be severe, especially in systems containing financial or personal data.

Command Injection occurs when an application passes unsafe input to system-level commands. Unlike SQL Injection, this vulnerability directly affects the operating system. Attackers may: Execute arbitrary system commands, Modify server files, Install malicious software, Gain remote shell access.

This type of injection is particularly dangerous because it can lead to full server compromise. It often appears in applications that use system calls to execute external programs without sanitizing input. ross-Site Scripting (XSS) is a form of injection that targets the client-side environment, typically web browsers. It allows attackers to inject malicious JavaScript code into web pages

viewed by other users. XSS is commonly classified into: Stored XSS: Malicious script permanently stored on the server , Reflected XSS: Script executed via manipulated URL or input, DOM-based XSS: Exploits client-side JavaScript vulnerabilities, Consequences of XSS include:Session hijacking, Credential theft, Defacement of websites , Redirection to malicious sites

XSS highlights that injection is not limited to servers but also affects client-side security.

Injection vulnerabilities arise due to multiple technical and organizational weaknesses:

- Direct concatenation of user input into queries
- Lack of input validation and sanitization
- Insecure software design patterns
- Absence of secure coding standards
- Insufficient developer knowledge in secure programming
- Lack of automated security testing in SDLC

In many cases, development teams prioritize functionality over security, which increases vulnerability exposure.

CONCLUSION AND RECOMMENDATIONS

Injection vulnerabilities remain one of the most critical threats to modern information systems due to their simplicity and high impact. These attacks exploit improper handling of user input, allowing attackers to execute malicious code within an application environment. Preventing injection requires a combination of secure coding practices, proper input validation, and continuous security testing. By adopting modern security frameworks and following best practices, organizations can significantly reduce the risk of injection-based attacks and protect sensitive data effectively.

REFERENCES

1. OWASP. *OWASP Top 10: The Ten Most Critical Web Application Security Risks*. OWASP Foundation, 2021.
2. National Institute of Standards and Technology. *Security and Privacy Controls for Information Systems and Organizations (SP 800-53)*. NIST, 2020.
3. William Stallings. *Cryptography and Network Security: Principles and Practice*. 7th Edition. Pearson, 2017.
4. Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2nd Edition. Wiley, 2008.
5. ISO. *ISO/IEC 27001: Information Security Management Systems*. ISO, 2013.
6. Andrew Hoffman. *Web Application Security: Exploitation and Countermeasures for Modern Web Applications*. O'Reilly Media, 2020.
7. Cloud Security Alliance. *Security Guidance for Critical Areas of Focus in Cloud Computing*. CSA, 2019.
8. IETF. *RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. 2014.

OWASP. *Access Control Cheat Sheet*. OWASP Foundation, 2023.