IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AUTOMATED WEB VULNERABILITY SCANNER

A Comprehensive Tool for Scalable Web Vulnerability Analysis

[1]B. Nitin, [2]Ch. Ajjhayy Sai, [3]D. Naveen Kumar,[4]K. Kamalakar

[1234] *Undergraduate* Students

Hyderabad Institute of Technology and Management, Medchal, Hyderabad, Telangana

Abstract: Protecting digital infrastructure and sensitive information has become a critical priority as the use of web applications and institutional networks expands. Automated vulnerability scanners play a vital role in efficiently identifying and addressing security gaps that may otherwise be overlooked by manual assessments. This project outlines the design of an Automated Web Vulnerability Scanner that combines static and dynamic techniques to uncover threats such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). Comprehensive reports and prioritized risk scores help users respond quickly to the most severe issues while reducing reliance on manual checks. The tool also supports cybersecurity education by giving learners practical insight into vulnerability detection, ultimately encouraging stronger security practices in both academic and organizational environments.

Index Terms - Vulnerability Scanner, Cyber security, Vulnerability Assessment, Network Security, Cyber security Awareness, Digital Security, Security Tools, Threat Mitigation, Vulnerability Management

I. INTRODUCTION

Rapid digital transformation has elevated cybersecurity to a vital concern for both individuals and organizations. As cyber-attacks and data breaches grow, protecting system integrity and sensitive data is increasingly important. Vulnerabilities in software and network configurations provide entry points for threats, making automated vulnerability scanners essential for proactive defense.

This work evaluates awareness and perceptions of vulnerability scanners within a college community, using surveys to gather insights from students, faculty, and IT staff.

The study explores the effectiveness of these tools and identifies gaps in cybersecurity knowledge and practices. By analyzing survey responses, the research highlights areas needing improvement in security education and training.

The developed automated scanner combines static and dynamic analysis alongside signature-based detection to identify weaknesses such as outdated software, misconfigurations, weak credentials, and insecure coding. Its reporting feature prioritizes risks and offers practical solutions for remediation.

Testing across diverse environments demonstrates the scanner's value in reducing exploitation risks and strengthening organizational security.

II. LITERATURE SURVEY

[1] Defending against Cross-Site Scripting Attacks—Lwin Khin Shar, Hee Beng Kuan Tan.

This study examines the causes and impact of Cross-Site Scripting (XSS) vulnerabilities, highlighting that insufficient input filtering and lack of user input validation in web applications developed with languages like PHP, Java, and .NET contribute to XSS risks. The paper demonstrates how unrestricted handling of user data leads to exploits that compromise application security. These insights underpin the importance of enforcing strict input controls and refining validation strategies, which directly influence the secure coding standards adopted in this project.

[2] Overcoming SQL Injection—Abdul Razzaq, Ali Hur, Sidra Shahbaz, Mudassar Masood, H Farooq Ahmad.

The authors investigate defense strategies for SQL injection, recognized as one of the most prevalent web application threats. The paper emphasizes the need for sound coding practices, vulnerability detection, and runtime protection methods for mitigating SQL injection risks. Their findings inspire the project's focus on integrating comprehensive detection modules and robust preventive mechanisms for parameterized queries and input sanitation.

[3] The Web Application Hacker's Guide: Finding and Exploiting Safety Flaws—Dafydd Stuttard, Marcus Pinto.
This book provides a detailed look into exploitable flaws found in web interfaces, explaining how attackers leverage these weaknesses to access personal information or disrupt normal functionality. Fully updated strategies and methodologies for attacking and defending modern web applications are presented, informing the architectural design of this project's vulnerability scanner, especially in supporting evolving security requirements.

[4] SQL Injection Attacks and Defense—Justin Clarke.
The work reviews the persistent dangers posed by SQL injection, describing it as a major threat for server-side applications. It highlights the need for dynamic defense mechanisms and demonstrates the limitations of language-specific analysis tools that may only detect basic vulnerabilities. These insights support the development of cross-platform and language-agnostic scanning capabilities within our system, removing dependency burdens and increasing coverage for diverse web applications.

Existing systems typically specialize in specific languages and only identify common vulnerabilities like XSS and SQL injection. This leads to increased complexity and costs, as tools are custom-fitted and may not be available for every environment. The generalized approach taken in this project addresses these challenges by providing a unified and comprehensive tool for vulnerability scanning, supporting broader language compatibility, and automating detection of both common and complex security flaws.

III. EXISTING METHOD

Current approaches to web vulnerability detection primarily leverage automated scanning tools that utilize a combination of static analysis, dynamic analysis, and signature-based techniques. These scanners inspect application source code, network configurations, and runtime behaviors to reveal security flaws such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and other common vulnerabilities.Static analysis focuses on reviewing code for insecure constructs or patterns known to facilitate attacks. Dynamic scanning simulates attacks on live applications by injecting payloads and monitoring responses for abnormal or unsafe behavior. Signature-based detection matches code and application characteristics against a database of known vulnerabilities, enhancing identification speed and efficiency for well-documented threats.
Despite widespread use, conventional scanners often concentrate on specific languages or platforms, limiting coverage for multi-language and complex web applications. Manual verification remains necessary for deeper security evaluation, as most tools rely on predefined rules and heuristics and may not detect zero-day or advanced exploits. Other challenges include reporting false positives/negatives, incomplete crawling of dynamic or single-page web interfaces, and dependency on external plugins, increasing time and resource requirements.
These limitations illustrate the need for advanced automated solutions capable of adapting to modern web architecture and evolving threat landscapes, motivating the development of integrated scanners that combine multiple analysis strategies and broader vulnerability coverage.

IV. PROPOSED METHOD

1. Aims to create a universal automated tool capable of scanning web applications for vulnerabilities regardless of the programming language they are written in.

2. This tool works by analysing the application through sending requests and scrutinizing the responses from the URL to detect cross-site scripting (XSS) vulnerabilities, and by examining the source code and providing recommendations for addressing SQL injection (SQLi) risks.

3.The results generated by this tool can then be integrated into the development process to enhance the quality of the code. By generating comprehensive vulnerability reports, this approach facilitates the identification of potential risks and attack avenues present on individual hosts.

4.1 LIMITATIONS OF PROPOSED METHOD

1. These systems often focus narrowly on certain languages, frameworks, or vulnerabilities and may fail to detect more sophisticated or zero-day risks.

2. Manual review is still required for deep analysis or to contextualize results, as automated scanners mostly detect issues that fit within predefined rulesets.

3. Inaccuracies can occur in reported vulnerabilities due to limitations in crawling depth, analysis scope, and application complexity (especially single-page applications).

4. Dependency on external libraries or plugins for specific platforms increases cost and time complexity

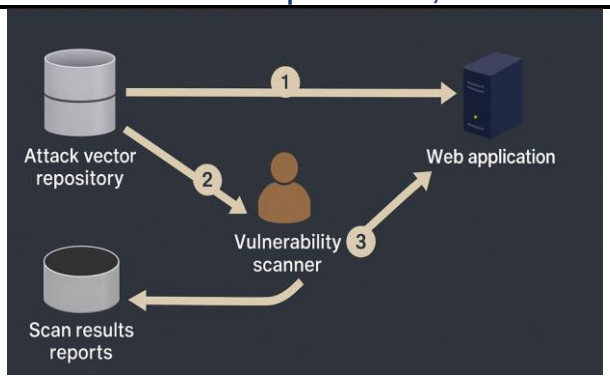

Fig.1

V. FLOW CHART

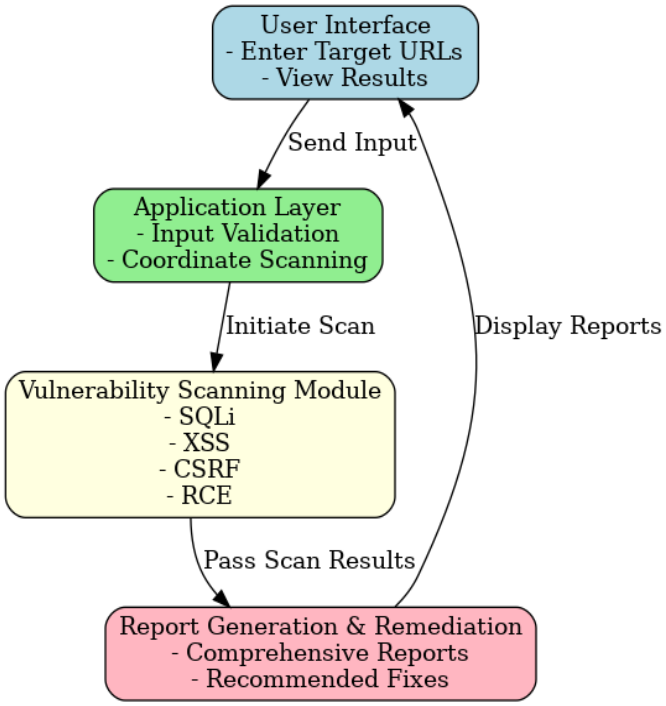

Fig.2

VI. WORKING MODEL OF PROJECT

The working model of the automated web vulnerability scanner involves crawling the target web application to map its pages and input points, followed by generating and injecting crafted attack payloads to test for security flaws like SQL injection and XSS. Responses are analyzed to detect anomalies, and a comprehensive report is generated detailing vulnerabilities, their severity, and recommendations for remediation.

6.1 Input Module

The system accepts URLs of web applications to be scanned. It allows users to specify target URLs or upload source code files for analysis**.**

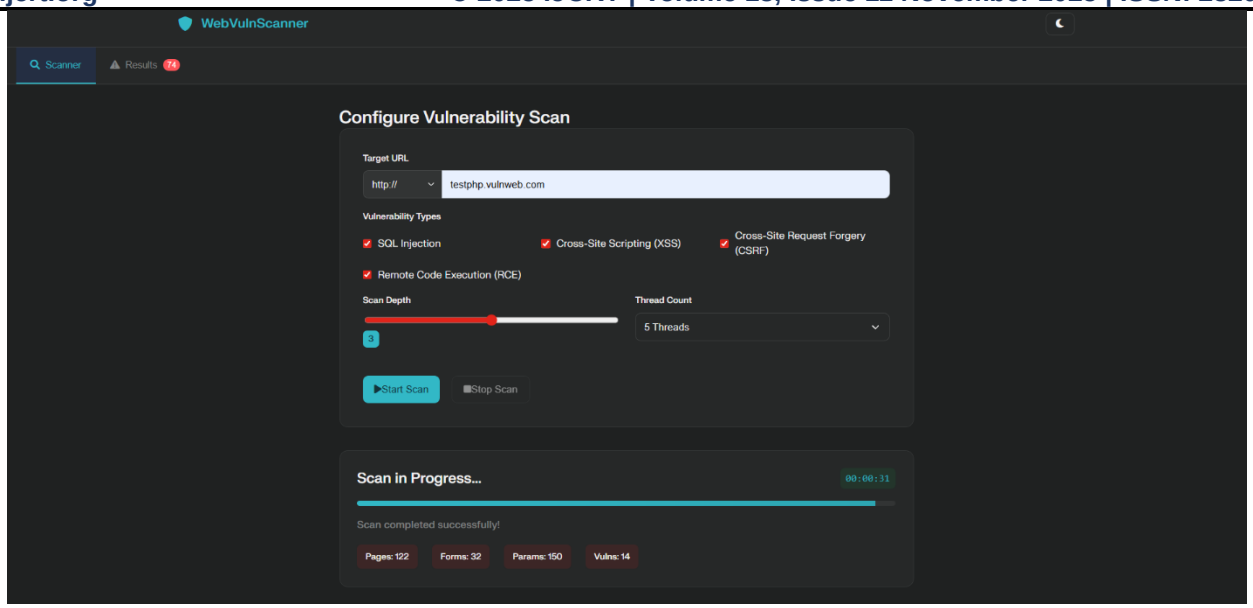

Fig.3

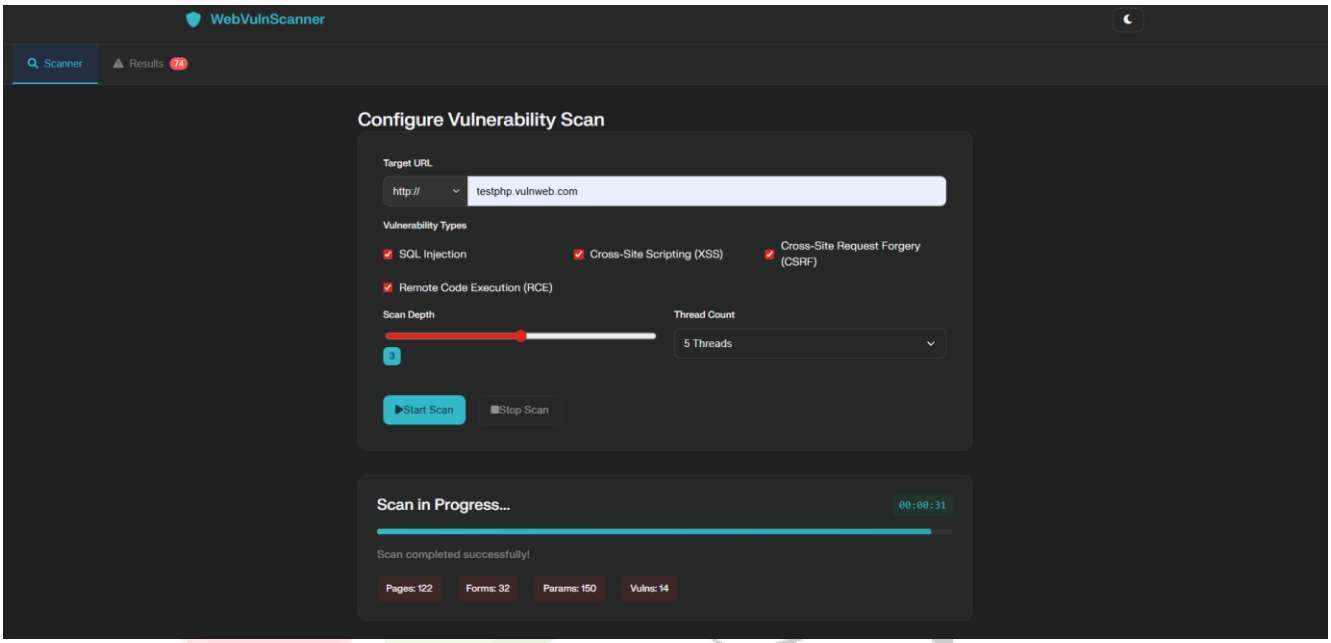

Fig.4

6.2 Scanner Module

Performs dynamic scanning by sending crafted requests and payloads to the application, interacting with JavaScript-heavy pages for thorough analysis.

6.3 Vulnerability Classification Module

Analyzes the responses and scan results to classify detected vulnerabilities by type and severity, such as SQL Injection, Cross-Site Scripting (XSS), etc.

6.4 Report Generation Module

Compiles detailed vulnerability reports, summarizing risks, affected components, and provides recommendations for mitigation, supporting integration with development workflows.

#	TYPE	SEVERITY	CVSS SCORE	LOCATION	DESCRIPTION	ACTIONS
1	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com	Standard SQL injection vulnerability allowing database manipulation	View Details
7	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/disclaimer.php	Standard SQL injection vulnerability allowing database manipulation	View Details
13	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/artists.php	Standard SQL injection vulnerability allowing database manipulation	View Details
19	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/index.php	Standard SQL injection vulnerability allowing database manipulation	View Details
25	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/cart.php	Standard SQL injection vulnerability allowing database manipulation	View Details
31	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/categories.php	Standard SQL injection vulnerability allowing database manipulation	View Details
41	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/guestbook.php	Standard SQL injection vulnerability allowing database manipulation	View Details
47	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/login.php	Standard SQL injection vulnerability allowing database manipulation	View Details
48	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/login.php	Standard SQL injection vulnerability allowing database manipulation	View Details
49	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/login.php	Standard SQL injection vulnerability allowing database manipulation	View Details
50	Classic SQL Injection	CRITICAL	9.3	http://testphp.vulnweb.com/login.php	Standard SQL injection vulnerability allowing database manipulation	View Details

Fig.5

VII. RESULT & CONCLUSION

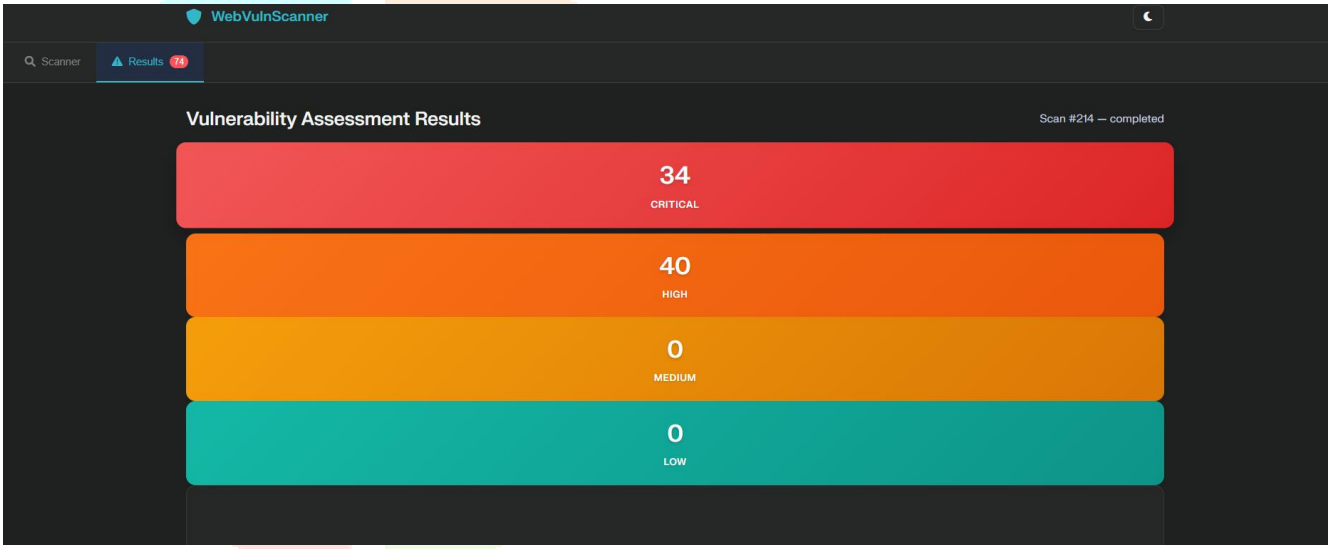

Fig.6

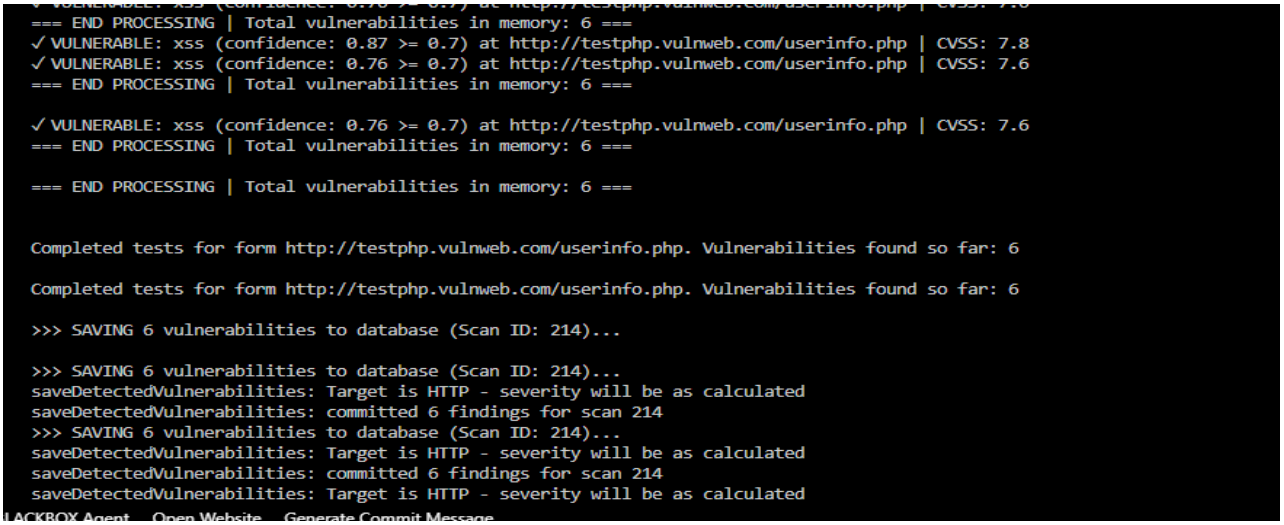

Fig.6

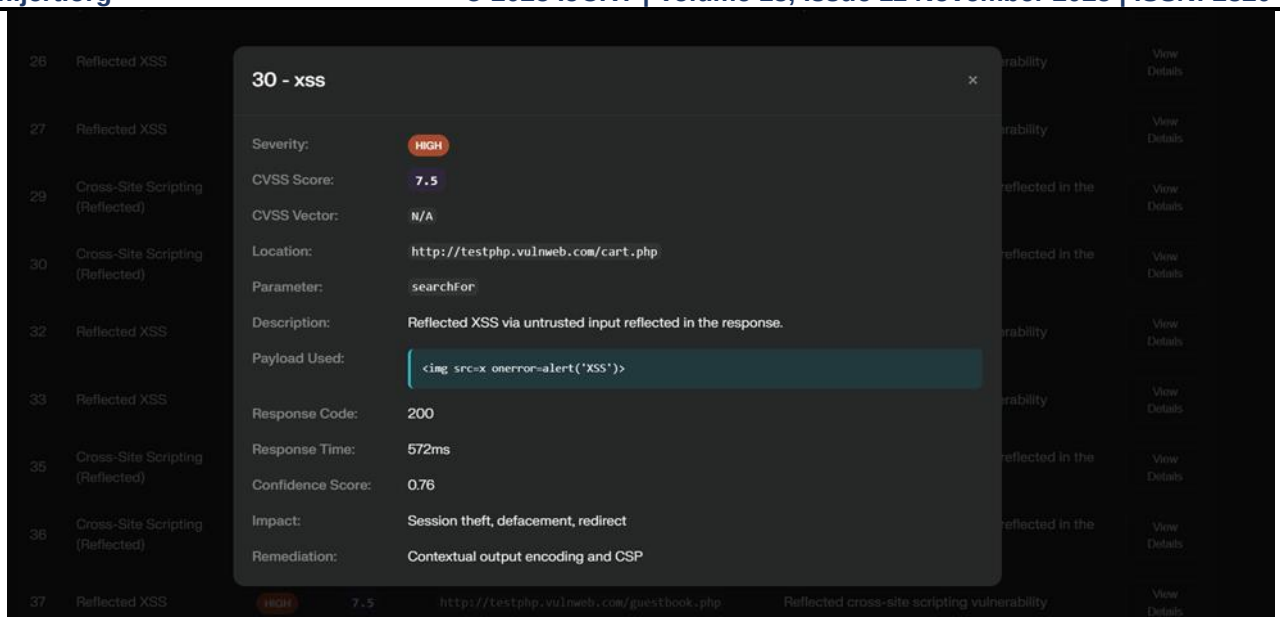

Fig.7

VIII. REFERENCES

[1] Shar, L. K. and Tan, H. B. K. 2012. Defending against Cross-Site Scripting Attacks. International Journal of Computer Science and Network Security, 12(1): 1-8.

[2] Razzaq, A., Hur, A., Shahbaz, S., Masood, M., and Ahmad, H. F. 2015. Overcoming SQL Injection. International Journal of Advanced Computer Science and Applications, 6(1): 123-129.

[3] Stuttard, D. and Pinto, M. 2011. The Web Application Hacker's Guide: Finding and Exploiting Safety Flaws. Wiley Publishing.

[4] Clarke, J. 2009. SQL Injection Attacks and Defense. Syngress