

Detection of Malicious Query Attack Weaknesses within Online Software Systems Using Byte-Level Pattern Matching

Dr. Ahmed Saeed Al-Mansoori

Department of Computer Engineering College of Engineering United Arab Emirates University Al Ain, United Arab Emirates

Article received: 16/02/2026, Article Accepted: 14/03/2026, Article Published: 16/04/2026

© 2026 Authors retain the copyright of their manuscripts, and all Open Access articles are disseminated under the terms of the [Creative Commons Attribution License 4.0 \(CC-BY\)](https://creativecommons.org/licenses/by/4.0/), which licenses unrestricted use, distribution, and reproduction in any medium, provided that the original work is appropriately cited.

ABSTRACT

Web-based software systems constitute the backbone of modern digital services, supporting financial transactions, healthcare systems, industrial control networks, and cloud-based infrastructures. However, the rapid expansion of online applications has significantly increased exposure to cybersecurity threats, particularly malicious query injection attacks targeting database-driven applications. These attacks exploit vulnerabilities in application-layer query processing mechanisms, allowing attackers to manipulate backend databases and retrieve or modify sensitive information. Traditional detection mechanisms such as rule-based filtering and signature-based intrusion detection systems often struggle to identify previously unseen attack patterns or obfuscated query manipulations. Consequently, there is a growing need for robust detection techniques capable of identifying structural vulnerabilities at deeper software layers.

This study proposes a novel approach for detecting malicious query attack weaknesses in online software systems through byte-level pattern matching techniques. Unlike conventional string-based detection methods, the proposed model examines compiled or intermediate representations of application components to identify structural similarities between known vulnerable patterns and target software binaries. The approach leverages byte-level similarity metrics, vulnerability signature mapping, and automated scanning processes to detect hidden weaknesses that may not be visible through source-level analysis.

The research integrates concepts from vulnerability assessment frameworks, binary similarity analysis, and modern cybersecurity monitoring models to develop an efficient vulnerability detection architecture. A comprehensive analysis of existing research in cybersecurity vulnerabilities, machine learning-based threat detection, and network security situation assessment provides the theoretical foundation for the proposed framework. The study further outlines a modular detection architecture composed of preprocessing, binary feature extraction, pattern similarity evaluation, and vulnerability classification components.

Experimental simulations demonstrate that byte-level similarity analysis can significantly improve the identification of malicious query attack patterns embedded within compiled application components. The results highlight improvements in detection accuracy, vulnerability coverage, and resilience against code obfuscation techniques compared to traditional source-code analysis approaches.

The findings contribute to the advancement of proactive vulnerability detection mechanisms for secure web application development. By integrating binary-level analysis with pattern matching techniques, the proposed model offers an effective method for strengthening software security infrastructures, reducing exploitation risks, and enhancing defensive strategies against evolving database-oriented cyber threats.

Keywords: Malicious query attacks, SQL injection vulnerabilities, binary code similarity, cybersecurity vulnerability detection, byte-level pattern matching, web application security, vulnerability analysis, intrusion detection, database security, software security assessment.

INTRODUCTION

The rapid evolution of internet technologies has transformed the way organizations manage information

systems, conduct transactions, and deliver digital services. Modern online software systems increasingly

rely on dynamic web applications connected to backend databases for storing, retrieving, and processing large volumes of data. While this architecture enables powerful functionality and real-time information processing, it also introduces critical security vulnerabilities that adversaries can exploit. Among the most prevalent threats targeting database-driven applications are malicious query injection attacks, which manipulate database queries to bypass authentication controls or access sensitive data.

Cybersecurity vulnerabilities in web applications represent a persistent challenge for organizations across multiple sectors, including healthcare, finance, industrial automation, and government infrastructures. Research on cybersecurity threats indicates that application-layer vulnerabilities remain one of the most exploited entry points for attackers attempting to compromise digital systems (Humayun et al., 2020). As web-based technologies become more complex and interconnected, the attack surface for potential vulnerabilities expands significantly, making effective detection and prevention strategies essential for maintaining system integrity.

Malicious query attacks, commonly associated with SQL injection techniques, exploit weaknesses in the interaction between user input validation mechanisms and backend database queries. When an application fails to properly sanitize input data, attackers can insert specially crafted query fragments that modify the intended behavior of database operations. Such attacks may allow unauthorized users to retrieve confidential information, manipulate stored records, or execute administrative database commands. These vulnerabilities can have severe consequences, including financial losses, data breaches, service disruptions, and reputational damage.

Traditional security approaches for detecting malicious query attacks typically rely on rule-based filtering systems, signature matching, or static code analysis. Although these techniques can identify known attack patterns, they often fail to detect advanced attack variations or obfuscated code structures. In many cases, attackers intentionally modify query structures or use encoding techniques to evade conventional detection mechanisms. Consequently, security researchers have increasingly explored alternative approaches that examine deeper layers of software architecture to identify hidden vulnerabilities.

Binary-level vulnerability analysis has emerged as a promising technique for identifying security weaknesses within compiled software components. Instead of analyzing source code directly, binary analysis examines compiled program instructions, machine-level representations, and execution patterns to detect similarities with known vulnerability signatures. This approach offers several advantages, including the ability

to analyze proprietary software without source code access and improved resilience against code obfuscation techniques.

Recent studies in cybersecurity research emphasize the importance of advanced detection techniques capable of identifying complex attack patterns in modern computing environments. For example, research on software vulnerability detection highlights the growing need for automated security assessment tools capable of analyzing large-scale code repositories and detecting hidden weaknesses before exploitation occurs (Ponta et al., 2020). Similarly, studies on cybersecurity threats and vulnerability management demonstrate that proactive detection strategies play a crucial role in reducing security risks in modern information systems (Jimmy, 2024).

In addition to software-level vulnerabilities, the proliferation of emerging technologies such as Internet of Things (IoT), cloud computing, and edge computing has introduced new attack vectors that further complicate cybersecurity management. IoT devices and cloud-based applications often interact with centralized databases through web interfaces, increasing the likelihood of query-based attacks targeting system backends (Mishra & Pandya, 2021). These technological developments highlight the necessity for innovative security mechanisms capable of identifying vulnerabilities across heterogeneous computing environments.

The primary objective of this research is to develop an advanced vulnerability detection framework capable of identifying malicious query attack weaknesses in online software systems using byte-level pattern matching techniques. By analyzing compiled application components and comparing their structural characteristics with known vulnerability patterns, the proposed approach aims to detect hidden security weaknesses that may not be identifiable through conventional source-code analysis.

The research focuses on three key objectives. First, it aims to investigate the limitations of existing vulnerability detection techniques in identifying malicious query attack patterns. Second, it proposes a binary similarity analysis framework designed to detect query-related vulnerabilities within compiled application modules. Third, the study evaluates the effectiveness of the proposed approach in improving vulnerability detection accuracy and enhancing software security assessment mechanisms.

The scope of this study is limited to the analysis of web-based application components that interact with relational databases and process dynamic user inputs. The proposed detection framework focuses specifically on identifying vulnerabilities associated with malicious query manipulation attacks. Although the methodology

may be applicable to other types of security vulnerabilities, the research primarily addresses database-oriented attack vectors.

This research contributes to the field of cybersecurity by introducing a novel vulnerability detection methodology that integrates binary code similarity analysis with byte-level pattern matching techniques. The findings aim to support software developers, cybersecurity analysts, and system administrators in strengthening the security of online software systems and reducing the risks associated with malicious query exploitation.

2. Literature Review

Cybersecurity vulnerabilities represent a fundamental challenge for modern digital infrastructures, particularly as software systems become increasingly interconnected and data-driven. A large body of research has focused on identifying, analyzing, and mitigating security vulnerabilities within web applications, cloud computing environments, and Internet of Things (IoT) ecosystems. The literature highlights several key themes related to vulnerability detection, security assessment, and threat mitigation strategies.

A comprehensive systematic mapping study on cybersecurity threats emphasizes that vulnerabilities in application-layer components are among the most common sources of security breaches (Humayun et al., 2020). The study categorizes security threats based on their origin, attack vector, and exploitation mechanism, demonstrating that database-related vulnerabilities remain a major concern for organizations deploying web-based systems. These vulnerabilities often arise due to improper input validation, insecure query construction, and inadequate authentication mechanisms.

Research on vulnerability detection within open-source software ecosystems provides valuable insights into the challenges of identifying security weaknesses in complex software systems. Ponta et al. (2020) conducted an empirical study examining vulnerability detection mechanisms for open-source dependencies, highlighting the importance of automated analysis tools in identifying potential security flaws. Their findings indicate that many software vulnerabilities remain undetected until they are exploited by attackers, emphasizing the need for proactive detection approaches.

The rapid adoption of cloud computing technologies has further intensified cybersecurity concerns related to software vulnerabilities. Studies on cloud security challenges reveal that distributed computing environments introduce new attack surfaces, particularly in applications interacting with remote databases and external services (Tabrizchi & Rafsanjani, 2020). Cloud-based systems often rely on web interfaces for managing resources, which increases exposure to malicious query

attacks and other injection-based threats.

In addition to cloud infrastructures, the expansion of IoT technologies has created new cybersecurity challenges related to device connectivity and data processing. Research on IoT security highlights numerous vulnerabilities associated with device communication protocols, authentication mechanisms, and data management processes (Harbi et al., 2021). Many IoT systems rely on web-based dashboards or API interfaces for monitoring and controlling devices, making them potential targets for query injection attacks.

Another important area of cybersecurity research focuses on the detection of vulnerabilities through machine learning and data-driven analysis techniques. Recent studies have explored the application of deep learning models for identifying abnormal patterns in network traffic and system behavior. For example, Duraibi and Alashjaee (2024) proposed a hybrid deep learning framework for detecting cyberattacks in IoT environments using dimensionality reduction techniques. Their findings demonstrate that machine learning approaches can significantly improve the detection of sophisticated cyber threats.

Similarly, research on adversarial machine learning highlights the increasing complexity of cyber threats targeting modern computing systems (Khan & Ghafoor, 2024). Attackers frequently use adversarial techniques to evade traditional security detection mechanisms, requiring the development of more advanced analytical approaches capable of identifying hidden vulnerabilities and attack patterns.

Network security monitoring and situational awareness systems also play a critical role in cybersecurity management. Several studies have proposed models for assessing network security conditions and predicting potential attack scenarios. For instance, Zhang et al. (2022) introduced a network security situation assessment model based on neural network optimization techniques, demonstrating how machine learning algorithms can be used to analyze network behavior and detect anomalies. Similarly, Zhao et al. (2023) proposed an attention-based prediction model for network security situation analysis, highlighting the role of intelligent monitoring systems in cybersecurity management.

Research on vulnerability detection within software packages further illustrates the challenges of identifying hidden security weaknesses. Alfadel et al. (2023) conducted an empirical analysis of vulnerabilities in Python packages, revealing that a significant number of security flaws originate from third-party dependencies. These vulnerabilities can propagate across multiple software systems, increasing the risk of exploitation by malicious actors.

Despite the extensive body of research on cybersecurity vulnerabilities and threat detection mechanisms, several gaps remain in the current literature. Many existing detection techniques focus primarily on source-code analysis or network traffic monitoring, which may not be sufficient for identifying vulnerabilities embedded within compiled software components. Furthermore, traditional vulnerability detection approaches often rely on predefined signatures or rule-based filtering mechanisms, limiting their effectiveness against novel or obfuscated attack patterns.

The literature suggests that binary-level analysis techniques may offer a promising solution for addressing these limitations. By examining compiled program instructions and identifying structural similarities between vulnerable code segments, binary similarity analysis can reveal hidden security weaknesses that may not be detectable through conventional analysis methods.

Therefore, this research builds upon existing cybersecurity studies by proposing a vulnerability detection framework that integrates byte-level pattern matching with binary similarity analysis to identify malicious query attack weaknesses in online software systems.

3. Research Methodology

The objective of this research is to design and evaluate a detection framework capable of identifying malicious query attack weaknesses within online software systems using byte-level pattern matching techniques. The methodology is based on a structured analytical approach that integrates vulnerability assessment theory, binary code analysis, and pattern similarity evaluation. Unlike traditional static code analysis methods that rely on source-level inspection, the proposed methodology examines compiled program structures to detect hidden security weaknesses associated with query manipulation attacks.

The research adopts a technical paper methodology in which a conceptual detection model is designed, implemented in a simulated environment, and evaluated through controlled experiments. The methodology consists of four major phases: vulnerability pattern collection, binary feature extraction, similarity evaluation, and vulnerability classification. Each phase contributes to the overall objective of identifying structural patterns that indicate the presence of malicious query attack weaknesses.

The first phase focuses on constructing a vulnerability pattern repository derived from known database attack structures and previously identified software vulnerabilities. Existing studies demonstrate that vulnerability databases and signature repositories are essential for automated security assessment systems

(Ponta et al., 2020). These repositories contain patterns representing insecure query constructions, improper input validation routines, and unsafe database interaction mechanisms.

The second phase involves extracting byte-level features from compiled software modules. Instead of analyzing source code, the proposed approach converts application components into intermediate binary representations. This allows the detection system to identify vulnerabilities even when source code is unavailable or obfuscated. Binary-level analysis has been shown to be effective in identifying hidden weaknesses in software packages and compiled libraries (Alfadel et al., 2023).

The third phase performs similarity evaluation between extracted binary patterns and known vulnerability signatures. Pattern matching algorithms are used to determine structural similarity based on instruction sequences, memory access patterns, and control-flow characteristics. Similarity-based detection methods have been widely applied in cybersecurity research for identifying abnormal behavior and hidden attack patterns (Duraibi & Alashjaee, 2024).

The final phase classifies detected similarities into vulnerability categories. The classification process determines whether the identified pattern corresponds to a malicious query attack weakness, a general software vulnerability, or a non-threatening code structure. This classification step enables the system to provide accurate vulnerability reports and reduce false positive detections.

The proposed methodology is designed to be independent of programming language, platform, or development framework. By operating at the byte level, the detection framework can analyze web applications developed using different technologies, including Java, Python, PHP, and .NET environments. This platform independence increases the applicability of the proposed model for real-world cybersecurity assessment.

4. Proposed Detection Framework

The proposed detection framework is designed to identify malicious query attack weaknesses in online software systems by analyzing compiled program structures using byte-level pattern matching. The framework integrates concepts from vulnerability assessment, binary similarity analysis, and automated security monitoring to create a multi-layer detection architecture.

The framework consists of four primary components: input preprocessing, binary feature extraction, pattern similarity engine, and vulnerability classification module. These components operate sequentially to analyze software binaries and detect potential attack weaknesses.

The input preprocessing module prepares application files for analysis. Online software systems often contain multiple executable modules, libraries, and scripts. The preprocessing stage collects all relevant components that interact with database queries or user inputs. These components are converted into binary form to allow low-level analysis.

The binary feature extraction module identifies structural characteristics within the compiled program. These characteristics include opcode sequences, control flow patterns, memory access instructions, and function call structures. Previous research has shown that binary-level features provide valuable information for identifying hidden vulnerabilities that may not appear in high-level source code (Alfadel et al., 2023).

The pattern similarity engine compares extracted binary features with known vulnerability signatures stored in the repository. The similarity evaluation process uses pattern matching algorithms to determine whether the structure of the analyzed code resembles known insecure query patterns. Similarity-based detection methods are commonly used in anomaly detection and intrusion detection systems (Zhao et al., 2023).

The vulnerability classification module interprets the similarity results and determines the severity of the detected weakness. If the similarity score exceeds a predefined threshold, the system marks the code segment as vulnerable. The classification module also categorizes vulnerabilities based on type, such as query injection weakness, authentication bypass risk, or data manipulation vulnerability.

The proposed framework supports automated scanning of large-scale software systems. This capability is essential because modern web applications often consist of thousands of code files and multiple external dependencies. Automated vulnerability detection reduces manual effort and improves security assessment efficiency.

5. Byte-Level Pattern Matching Model

Byte-level pattern matching is the core technique used in the proposed detection framework. Unlike traditional text-based analysis, byte-level matching examines compiled instructions directly, allowing the detection system to identify structural similarities even when the code is modified or obfuscated.

A byte-level pattern represents a sequence of machine instructions that corresponds to a specific vulnerability structure. For example, an insecure query execution routine may produce a characteristic sequence of instructions related to input reading, string concatenation, and database command execution. By comparing these instruction sequences with known vulnerability patterns,

the detection system can identify potential attack weaknesses.

Pattern matching is performed using similarity metrics that measure how closely two byte sequences resemble each other. Common similarity metrics include sequence alignment, hash-based comparison, and control-flow similarity analysis. These techniques allow the system to detect vulnerabilities even when small modifications are present.

The use of byte-level analysis provides several advantages over source-code inspection. First, it allows vulnerability detection in proprietary software where source code is not available. Second, it reduces the impact of code obfuscation techniques often used by attackers. Third, it enables detection across multiple programming languages because all compiled programs ultimately produce machine-level instructions.

Research in cybersecurity vulnerability detection indicates that structural similarity analysis can significantly improve detection accuracy for complex attack patterns (Khan & Ghafoor, 2024). Therefore, byte-level pattern matching provides a reliable foundation for identifying malicious query attack weaknesses.

6. Binary Similarity Evaluation Algorithm

The binary similarity evaluation algorithm determines whether a compiled program contains patterns that resemble known malicious query attack structures. The algorithm operates in several steps.

First, the binary file is divided into instruction blocks. Each block represents a sequence of machine-level operations related to a specific function or procedure. These blocks are extracted using disassembly techniques.

Second, each instruction block is converted into a feature vector representing opcode frequency, instruction order, and control-flow structure. Feature vectors allow efficient comparison between different code segments.

Third, similarity scores are calculated between extracted feature vectors and vulnerability patterns stored in the repository. The similarity score is computed using a weighted comparison formula that considers opcode similarity, sequence alignment, and structural matching.

Fourth, the algorithm compares the similarity score with a predefined threshold. If the score exceeds the threshold, the code block is marked as potentially vulnerable. Otherwise, it is considered safe.

Fifth, detected vulnerabilities are grouped into categories based on pattern type. This classification allows the system to identify specific attack weaknesses related to malicious query execution.

Similarity-based algorithms have been widely used in cybersecurity monitoring systems to detect abnormal behavior and hidden threats (Yu, 2024). The proposed algorithm adapts these techniques for vulnerability detection within compiled web application components.

7. System Architecture

The architecture of the proposed detection system follows a modular design to ensure flexibility, scalability, and efficiency. The system consists of five major layers: application input layer, binary analysis layer, similarity processing layer, vulnerability evaluation layer, and reporting layer.

The application input layer collects executable files, libraries, and scripts from the target software system. This layer ensures that all components interacting with database queries are included in the analysis.

The binary analysis layer converts input files into machine-level representations. Disassembly tools and binary parsers are used to extract instruction sequences from compiled programs. This layer prepares the data for similarity evaluation.

The similarity processing layer performs byte-level pattern matching between extracted instruction sequences and known vulnerability signatures. This layer uses similarity algorithms to identify structural matches.

The vulnerability evaluation layer analyzes similarity results and determines whether the detected pattern represents a malicious query attack weakness. The evaluation process considers similarity score, pattern type, and execution context.

The reporting layer generates vulnerability reports describing detected weaknesses, severity level, and recommended mitigation strategies. Automated reporting is essential for large-scale software security assessment.

Modern cybersecurity systems require multi-layer architectures to handle complex threat detection tasks (Mughal, 2022). The proposed architecture follows this principle to ensure accurate and efficient vulnerability detection.

8. Experimental Setup

To evaluate the effectiveness of the proposed detection framework, a simulated environment was created consisting of multiple web-based software modules interacting with relational databases. The experimental setup included applications developed using different programming languages to verify the platform independence of the byte-level detection approach.

Known vulnerable code samples were intentionally included in the test environment to represent malicious

query attack weaknesses. These samples contained insecure query constructions, improper input validation routines, and unsafe database execution methods.

The vulnerability repository was constructed using previously identified security patterns from software vulnerability studies and cybersecurity research. These patterns were converted into byte-level signatures for similarity comparison.

The detection framework was then executed to analyze the compiled application components. The system recorded detection accuracy, similarity scores, false positive rate, and processing time.

The experimental evaluation focused on measuring how effectively the proposed method detects malicious query attack weaknesses compared to traditional static analysis techniques.

9. Results

The experimental evaluation of the proposed byte-level pattern matching framework was conducted to measure its effectiveness in detecting malicious query attack weaknesses in online software systems. The performance of the model was analyzed using several metrics, including detection accuracy, false positive rate, similarity score reliability, and processing efficiency. The results were compared with conventional static analysis techniques that rely on source-level inspection and rule-based vulnerability detection.

The proposed framework successfully detected the majority of malicious query attack patterns embedded in the test applications. In the experimental environment, vulnerable modules containing insecure query execution routines were correctly identified with high similarity scores. The average detection accuracy observed during testing exceeded that of traditional signature-based detection methods, particularly in cases where the source code had been modified or partially obfuscated. This confirms that byte-level pattern matching provides a deeper level of analysis compared to text-based inspection.

One of the most significant findings was the ability of the system to detect vulnerabilities even when query structures were altered at the source level. In several test cases, insecure query patterns were hidden through variable renaming, code restructuring, or encoding techniques. Conventional static analysis tools failed to recognize these vulnerabilities because the textual representation of the query had changed. However, the compiled binary structure remained similar, allowing the proposed method to identify the vulnerability through instruction-level similarity comparison.

The false positive rate of the system remained within

acceptable limits during testing. Although some safe code segments produced partial similarity matches with known vulnerability patterns, the classification module correctly filtered most of these cases using threshold-based evaluation. This demonstrates that combining similarity scoring with vulnerability classification improves the reliability of detection results.

Processing efficiency was also evaluated to determine the feasibility of applying the method to large-scale software systems. The results showed that the binary similarity algorithm required more processing time than simple rule-based scanning but remained practical for automated security assessment. Optimization techniques such as feature vector indexing and block-level comparison reduced the computational overhead and allowed the framework to analyze large applications within reasonable time limits.

Another important observation was the framework's ability to detect vulnerabilities in applications developed using different programming languages. Since the analysis operates at the byte level, the system was able to examine compiled modules produced from Java, Python, and C-based environments without requiring language-specific rules. This confirms the platform independence of the proposed detection model.

Overall, the experimental results indicate that byte-level pattern matching significantly enhances the detection of malicious query attack weaknesses in online software systems. The method demonstrates improved resilience against code obfuscation, higher detection accuracy, and better coverage of hidden vulnerabilities compared to traditional approaches.

10. Discussion

The results of this study demonstrate that analyzing compiled program structures using byte-level pattern matching provides a reliable method for detecting malicious query attack weaknesses in online software systems. The proposed framework addresses several limitations of traditional vulnerability detection techniques, particularly those related to source-code dependency and signature-based filtering. By focusing on binary-level similarity, the model is capable of identifying structural patterns that remain consistent even when the source code is modified or intentionally obfuscated.

One of the major advantages of the proposed approach is its ability to detect vulnerabilities in environments where source code is unavailable. Many commercial and proprietary applications are distributed only in compiled form, making conventional static analysis difficult or impossible. Binary similarity analysis allows security analysts to evaluate such systems without requiring access to the original code. This capability is consistent

with previous research indicating that binary-level vulnerability detection can improve security assessment in complex software ecosystems (Alfadel et al., 2023).

The findings also support the argument that modern cybersecurity threats require more advanced detection mechanisms than traditional rule-based systems. Attackers frequently use obfuscation, encoding, and structural modification techniques to bypass signature-based detection tools. As discussed in previous cybersecurity studies, adaptive and similarity-based analysis methods are necessary to detect evolving attack patterns (Khan & Ghafoor, 2024). The proposed framework aligns with this requirement by comparing structural characteristics rather than relying only on textual patterns.

Another important implication of the results is the applicability of the proposed model to emerging computing environments such as cloud systems, IoT networks, and distributed applications. These environments often involve multiple layers of software interacting with databases through web interfaces, increasing the risk of malicious query attacks. Research on cloud and IoT security highlights the growing need for automated vulnerability detection methods capable of analyzing heterogeneous software components (Tabrizchi & Rafsanjani, 2020; Mishra & Pandya, 2021). The byte-level approach used in this study is well suited for such environments because it is independent of programming language and development platform.

Despite the advantages, the proposed framework also has several limitations. Binary-level analysis requires additional processing time compared to simple static scanning, which may affect performance when analyzing very large systems. Furthermore, similarity-based detection depends on the quality of the vulnerability pattern repository. If the repository does not contain representative patterns, some vulnerabilities may remain undetected. Future research should focus on improving pattern databases and incorporating machine learning techniques to enhance pattern recognition accuracy.

Another limitation is that the current study focuses primarily on malicious query attack weaknesses related to database interactions. Although the methodology can be extended to other types of vulnerabilities, additional pattern definitions would be required to detect different attack categories. Expanding the framework to support multiple vulnerability types would increase its usefulness for comprehensive security assessment.

Overall, the discussion confirms that byte-level pattern matching provides a strong foundation for advanced vulnerability detection systems. The approach complements existing security tools and can be integrated into automated software testing and cybersecurity monitoring frameworks to improve protection against

database-oriented attacks.

11. Conclusion

This research presented a byte-level pattern matching framework for detecting malicious query attack weaknesses in online software systems. The study addressed the limitations of traditional vulnerability detection methods by proposing an approach that analyzes compiled program structures instead of relying solely on source-level inspection. The proposed framework integrates binary feature extraction, similarity evaluation, and vulnerability classification to identify hidden attack patterns within application components.

The experimental results demonstrated that the proposed method improves detection accuracy and provides stronger resistance against code obfuscation techniques. The ability to analyze compiled modules makes the framework suitable for modern software environments where source code may not be accessible. The study also showed that the approach can be applied to applications developed using different programming languages, increasing its practicality for real-world cybersecurity assessment.

The findings contribute to the field of software security by introducing a detection methodology that combines binary similarity analysis with vulnerability pattern matching. This approach supports proactive identification of security weaknesses before they can be exploited by attackers. In addition, the modular architecture of the framework allows integration with existing security monitoring and vulnerability management systems.

Future research should focus on expanding the vulnerability pattern repository, improving similarity algorithms, and incorporating machine learning techniques to enhance detection performance. Additional studies may also evaluate the framework in large-scale industrial environments and extend the model to detect other types of cyber threats beyond malicious query attacks.

Strengthening vulnerability detection mechanisms remains essential for protecting modern online software systems. The proposed byte-level pattern matching approach provides a promising direction for developing more reliable and efficient cybersecurity assessment tools.

REFERENCES

1. Alfadel, M., Costa, D. E., and Shihab, E. (2023). Empirical analysis of security vulnerabilities in python packages. *Empirical Software Engineering*, 28 (3), 59–70.
2. Duraibi, S. and Mujawib Alashjaee, A. (2024). Enhancing Cyberattack Detection Using Dimensionality Reduction With Hybrid Deep Learning on Internet of Things Environment. *IEEE Access*, vol. 12, pp. 84752–84762.
3. Harbi, Y., Aliouat, Z., Refoufi, A., and Harous, S. (2021). Recent security trends in internet of things: A comprehensive survey. *IEEE Access*, 9 (1), 113292–113314.
4. Humayun, Mamoon, Mahmood Niazi, N. Z. Jhanjhi, Mohammad Alshayeb, and Sajjad Mahmood. *Cyber security threats and vulnerabilities: a systematic map study*. *Arabian Journal for Science and Engineering* 45 (2020): 3171–3189.
5. Jiang, X., Lora, M., and Chattopadhyay, S. (2020). An experimental analysis of security vulnerabilities in industrial IoT devices. *ACM Transactions on Internet Technology (TOIT)*, 20 (2), 1–24.
6. Jimmy, F. N. U. (2024). Cyber security Vulnerabilities and Remediation Through Cloud Security Tools. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, 2 (1), 129–171.
7. Khan, M., and Ghafoor, L. (2024). Adversarial Machine Learning in the Context of Network Security: Challenges and Solutions. *Journal of Computational Intelligence and Robotics*, 4 (1), 51–63.
8. Kim, D. W., Choi, J. Y., and Han, K. H. (2020). Risk management-based security evaluation model for telemedicine systems. *BMC medical informatics and decision making*, 20 (1), 1–14.
9. Luan, M., B. Wang, Y. Zhao, and F. Hu. (2021). Anomalous Subgraph Detection in Given Expected Degree Networks With Deep Learning. *IEEE Access*, vol. 9, pp. 60052–60062.
10. Mishra, N., and Pandya, S. (2021). Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, 9 (1), 59353–59377.
11. Mrabet, H., Belguith, S., Alhomoud, A., and Jemai, A. (2020). A survey of IoT security based on a layered architecture of sensing and data analysis. *Sensors*, 20 (13), 3625–3637.
12. Mughal, A. A. (2022). Building and Securing the Modern Security Operations Center (SOC). *International Journal of Business Intelligence and Big Data Analytics*, 5 (1), 1–15.
13. Ponta, S. E., Plate, H., and Sabetta, A. (2020).

- Detection, assessment and mitigation of vulnerabilities in open source dependencies. *Empirical Software Engineering*, 25 (5), 3175–3215.
14. Ranaweera, P., Jurcut, A., and Liyanage, M. (2021). MEC-enabled 5G use cases: a survey on security vulnerabilities and countermeasures. *ACM Computing Surveys (CSUR)*, 54 (9), 1–37.
15. Rawat, R., Chakrawarti, R. K., Vyas, P., Gonzales, J. L. A., Sikarwar, R., and Bhardwaj, R. (2023). Intelligent fog computing surveillance system for crime and vulnerability identification and tracing. *International Journal of Information Security and Privacy (IJISP)*, 17 (1), 1–25.
16. Somasundaram, R., and Thirugnanam, M. (2021). Review of security challenges in healthcare internet of things. *Wireless Networks*, 27 (8), 5503–5509.
17. Tabrizchi, H., and Kuchaki Rafsanjani, M (2020). A survey on security challenges in cloud computing: issues, threats, and solutions. *The journal of supercomputing*, 76 (12), 9493–9532.
18. Tawalbeh, L. A., Muheidat, F., Tawalbeh, M., and Quwaider, M. (2020). IoT Privacy and security: Challenges and solutions. *Applied Sciences*, 10 (12), 4102–4112.
19. Vyas, B. (2023). Security Challenges and Solutions in Java Application Development. *Eduzone: International Peer Reviewed/Refereed Multidisciplinary Journal*, 12 (2), 268–275.
20. Yaacoub, J. P. A., Noura, H. N., Salman, O., and Chehab, A. (2022). Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations. *International Journal of Information Security*, 21 (1), 115–158.
21. Yang, H., R. Zeng, G. Xu, and L. Zhang. (2021). A network security situation assessment method based on adversarial deep learning. *Applied Soft Computing*, vol. 102, 2021.
22. Yuan, L. (2021). Prediction of network security situation awareness based on an improved model combined with neural network. *Security and Privacy*, vol. 4, no. 6, 2021.
23. Yu, M., Zhuge, J., Cao, M., Shi, Z., and Jiang, L. (2020). A survey of security vulnerability analysis, discovery, detection, and mitigation on IoT devices. *Future Internet*, 12 (2), 27–36.
24. Yu, G.-F. (2024). A multi-objective decision method for the network security situation grade assessment under multi-source information. *Information Fusion*, vol. 102, 2024.
25. Zhang, R., Z. Pan, Y. Yin, and Z. Cai. (2022). A Model of Network Security Situation Assessment Based on BPNN Optimized by SAA-SSA. *International Journal of Digital Crime and Forensics*, vol. 14, no. 2, 2022.
26. Zhang, R., M. Liu, Z. Pan, and Y. Yin. (2022). Network Security Situation Assessment Based on Improved WOA-SVM. *IEEE Access*, vol. 10, pp. 96273–96283, 2022.
27. Zhao, D., G. Ji, and S. Zeng. (2023). Network security situation assessment based on dual attention mechanism and HHO-ResNeXt. *Connection Science*, vol. 35, no. 1, 2023.
28. Zhao, D., P. Shen, and S. Zeng. (2023). ALSNAP: Attention-based long and short-period network security situation prediction. *Ad Hoc Networks*, vol. 150, 2023.
29. Zhu, Y., and Z. Du. (2021). Research on the Key Technologies of Network Security-Oriented Situation Prediction. *Scientific Programming*, vol. 2021, 2021.
30. Wu, Y.-X., and D.-M. Zhao. (2024). Build IPSO-ABiLSTM Model for Network Security Situation Prediction. *Journal of Information Science and Engineering*, vol. 40, no. 1, pp. 71–88, 2024.