Enhancing Web Application Firewalls with Proxy Grammar for SQL Injection Defense

SHAIK SAFIYA

PG Scholar, Student, Department of CSE, Srinivasa Institute Of Technology And Science

Kadapa,Andhra Pradesh

K CHANDRA PRASAD

Assistant Professor, Department of CSE, Srinivasa Institute Of Technology And Science
Kadapa,Andhra Pradesh

A RAVI SANKAR

Associate Professor & HOD, Department of CSE, Srinivasa Institute Of Technology And Science
Kadapa,Andhra Pradesh

1 2 3

Abstract: Web applications are inherently vulnerable due to their open nature, making their security critical for organizations of all sizes. These platforms often store sensitive data, and any exploitation of their vulnerabilities—especially through SQL injection (SQLi) attacks—can lead to severe data breaches, financial loss, and reputational damage. SQLi remains one of the most common and dangerous forms of attack used by malicious actors to compromise web application security. Web Application Firewalls (WAFs) serve as a frontline defense against such threats. Although recent research has introduced several advancements in WAF technologies to prevent SQLi, many solutions fall short by only evaluating WAF effectiveness without providing mechanisms to patch identified vulnerabilities. Others offer patches limited to the specific syntax supported by the tested WAF. To address these limitations, this paper presents PROGESI (PROxy Grammar to Enhance SQL Injection prevention)—a novel framework designed to reinforce WAF capabilities. PROGESI functions as an intermediary layer between a web server and incoming application-level requests. It can be deployed either independently or alongside existing WAFs and applies a set of grammar-based rules to dynamically patch SQLi vulnerabilities identified in the target server. Notably, PROGESI also detects and mitigates SQLi attempts that involve mutation techniques, thanks to its generalized rule-based approach. Experimental results demonstrate two key strengths of PROGESI: (i) enhanced detection of SQLi attacks—even in the presence of existing server-side defenses—with effectiveness improving as rule generalization increases; and (ii) superior detection accuracy, outperforming state-of-the-art methods even at lower levels of generalization, as validated on a benchmark SQLi dataset.

Keywords: Ethereum, phishing scams, blockchain security, phishing gang detection, transaction analysis,

I. INTRODUCTION

Web applications have been integral to business operations for over two decades, serving as the interface for various enterprise functions. Most of these applications are backed by databases, which remain the predominant storage solution in organizational environments. However, their integration with web applications also introduces significant security risks, primarily due to exploitable vulnerabilities that can compromise sensitive data. One of the most persistent and severe threats is the SQL injection (SQLi) attack, which continues to grow in both frequency and impact [2], [14],. Common defenses against SQLi include input validation functions, web application firewalls (WAFs), and the use of prepared statements. Validation and WAFs aim to sanitize user input at the application level, while prepared statements bind user inputs to predefined query structures. Despite these measures, less common techniques have emerged, such as query behavior monitoring, which block anomalous SQL queries based on model deviations. However, these methods often operate without full awareness of how queries are interpreted by the underlying database management system (DBMS) [5], [6], [17].

A major challenge lies in the oversimplified or incorrect assumptions developers make about the interaction between server-side code and the DBMS. For instance, developers often assume that PHP's mysql_real_escape_string is sufficient to neutralize all potential SQLi payloads, which is not the case. Another common oversight is neglecting to revalidate data retrieved from the database before incorporating it into new queries, resulting in second-order injection vulnerabilities. For example, input like admin' -- may initially be sanitized, but if the DBMS stores and later retrieves it without adequate checks, the payload can still trigger an injection when reused. These vulnerabilities stem from a semantic mismatch between a developer's expectation of how SQL queries are processed and the DBMS's actual behavior. This mismatch weakens existing security mechanisms, potentially allowing SQLi attacks to succeed despite implemented defenses.

To mitigate this issue, a shift in defense strategy is proposed—moving the protection mechanisms inside the DBMS itself. By doing so, SQLi detection occurs after inputs are processed by server-side code and just before execution by the DBMS, reducing reliance on potentially flawed assumptions. This approach is inspired by similar in-process security strategies successfully applied in binary application protection, such as address space layout randomization

(ASLR), data execution prevention (DEP), and stack canaries [18], [21].

In this paper, we propose SEPTIC (SElf-ProtecTIng Databases from attaCks), a novel framework that embeds runtime protection directly within the DBMS. The DBMS is uniquely positioned to detect such attacks, given its definitive knowledge of SQL semantics, including clauses, predicates, and expressions—insight that external mechanisms lack.

SEPTIC addresses two primary classes of attacks: traditional SQLi and stored injection attacks, including stored cross-site scripting (XSS), both of which involve SQL queries. For SQLi, SEPTIC detects malicious queries by comparing them against a set of previously validated query models and using similarity-based matching to improve detection accuracy. For stored injections, it uses specialized plugins that sanitize or reject harmful data before it is committed to the database.

II. LITERATURE SURVEY

Web application attacks have been addressed through various mitigation strategies in the literature, which can be broadly classified based on the application layer they target. These include: (i) security by design approaches, which aim to prevent vulnerabilities from the initial stages of application development; (ii) machine learning models that predict and detect web-based attacks; and (iii) Web Application Firewalls (WAFs) that safeguard web servers from malicious HTTP traffic. This section focuses on security by design techniques, followed by a discussion of the motivation behind the proposed contribution.

Security by design represents a proactive defense mechanism in web development. These approaches aim to eliminate or reduce vulnerabilities—such as SQL injection (SQLi)—during the design and coding phases. One widely used method involves input sanitization, which restricts input to a predefined acceptable domain. If user input falls outside this domain, the application halts execution, effectively mitigating potential threats. Developers typically define rules for acceptable inputs rather than attempting to account for every possible malicious input .

In [14], a MySQL plugin named SQLBlock was proposed to protect popular PHP-based web applications from SQLi attacks. Another widely adopted secure design framework is Laravel [15], [16], which defends against SQLi by using bind variables—values passed to SQL queries through parameterized statements rather than as raw literals. This is implemented via Laravel's Eloquent ORM [17], which abstracts SQL interactions and encourages secure database access practices.

However, design-level protections should not be tied to specific programming languages. For instance, Java's prepared statement interfaces represent a language-dependent solution. To address this limitation, the authors in [18] explored language-independent secure design patterns, such as the strategy design pattern, where algorithms are selected dynamically to mitigate SQLi threats. Similarly, [19] extended the factory design pattern, implementing a secure function that leverages the libinjection library to detect and respond to SQLi attacks effectively.

Another alternative is the use of stored procedures [15], which encapsulate SQL logic within predefined subroutines. This adds an abstraction layer between user input and database execution, enhancing security by limiting direct SQL interactions.

In [15], an ad hoc design pattern is proposed specifically for lateral-based SQLi attacks. This pattern introduces an architectural model comprising three key zones: the injection zone, where user inputs are handled; the secure zone, which triggers the appropriate security mechanisms; and the sensitive zone, which protects critical data. This decoupling of user interaction and security logic enhances modularity while maintaining strong protection.

In summary, security by design remains a foundational approach for preventing SQLi and other injection-based attacks. By incorporating robust input validation, secure programming patterns, and architectural abstractions early in development, applications can significantly reduce their exposure to vulnerabilities..

3. METHODOLOGY

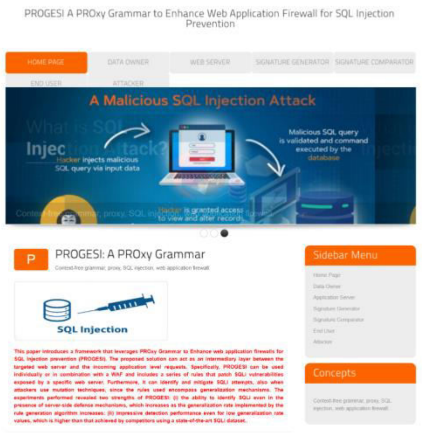
The proposed methodology introduces a layered defense framework designed to enhance SQL injection (SQLi) prevention in web applications by integrating a proxy grammar-based system, PROGESI, with existing Web Application Firewalls (WAFs). This framework operates as an intermediary layer between the client and the web server, intercepting and analyzing incoming HTTP requests. First, the system extracts and parses SQL queries from the requests using a domain-specific proxy grammar that models legitimate query patterns. A rule-based engine, enriched with generalization mechanisms, is then applied to detect deviations that signify SQLi attempts, including obfuscated or mutated attacks. The framework supports two operational modes: standalone—where PROGESI functions independently—and integrated—where it augments the capabilities of traditional WAFs. Furthermore, when an attack pattern is detected that bypasses existing WAF rules, PROGESI automatically generates patch rules that are specific to the affected server context. These patches are reusable and not bound to any programming language or SQL dialect, promoting broad applicability. To ensure adaptability, the framework supports rule aging and quarantine mechanisms to update detection logic dynamically in response to application updates or new attack vectors. This methodology emphasizes minimal false positives, strong generalization, and seamless deployment without altering the original application codebase.

IV. PROPOSED SYSTEM

With respect to security by design strategies, although Laravel includes features to protect against these SQL injection vulnerabilities (such as wrapping column names), some DB engines may still be vulnerable because they do not support binding variables (depending on their versions and configurations). Regarding the input sanitizers, this mitigation strategy is not applicable in every case because it could lead to the need for a complete rewrite of the source code software, which in some cases is not a cheap solution. In addition, inhibited characters may be present in some strings. Finally, the introduction of sophisticated input sanitizer methods could result in a degradation in web application performance. Although the efficiency of ML-based detection algorithms proves to be the most widespread and popular, the emerging cybersecurity frontiers of AI systems make these systems susceptible to adversarial attacks that can evade them..

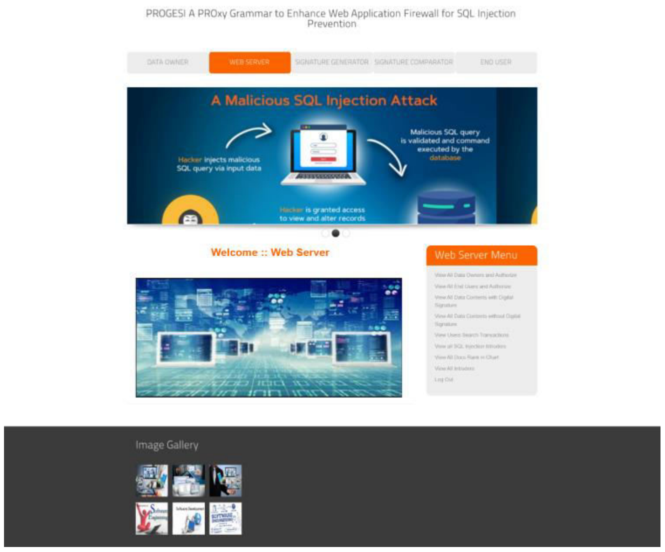
V.RESULTS

The screenshots of various phases of project are as follows

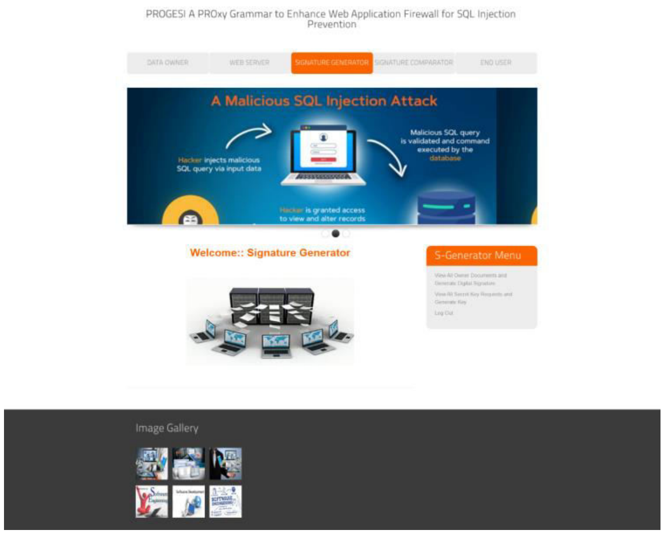

PROGESI A PROxy Grammar to Enhance Web Application Firewall for SQL Injection Prevention
HOME PAGE
DATA OWNER
WEB SERVER
SIGNATURE GENERATOR
SIGNATURE COMPARATOR
END USER
ATTACKER
A Malicious SQL Injection Attack
PROGESI: A PROxy Grammar
Context-free grammar, proxy, SQL injection, web application firewall
SQL Injection
Sidebar Menu
Concepts

This paper introduces a framework that leverages PROxy Grammar to Enhance web application firewalls for SQL injection prevention (PROGESI). The proposed solution can act as an intermediary layer between the targeted web server and the incoming application level requests. Specifically, PROGESI can be used individually or in combination with a WAF and includes a series of rules that patch SQLi vulnerabilities exposed by a specific web server. Furthermore, it can identify and mitigate SQLi attempts, also when attackers use mutation techniques, since the rules used encompass generalization mechanisms. The experiments performed revealed two strengths of PROGESI: (i) the ability to identify SQLi even in the presence of server-side defense mechanisms, which increases as the generalization rate implemented by the rule generation algorithm increases; (ii) impressive detection performance even for low generalization rate values, which is higher than that achieved by competitors using a state-of-the-art SQLi dataset.

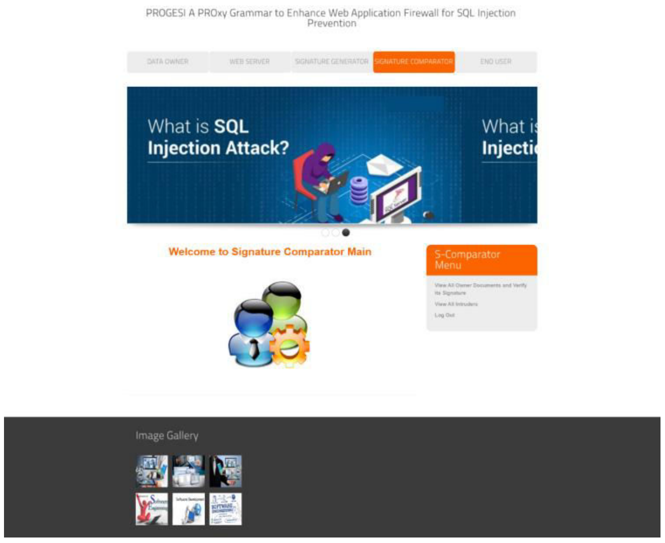
Screen 1:Home Page


PROGESI A PROxy Grammar to Enhance Web Application Firewall for SQL Injection Prevention
DATA OWNER
WEB SERVER
SIGNATURE GENERATOR
SIGNATURE COMPARATOR
END USER
A Malicious SQL Injection Attack
Welcome :: Web Server
Web Server Menu
Image Gallery

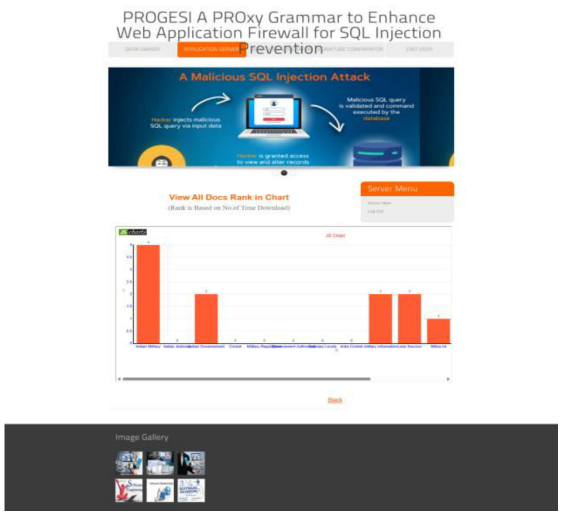
Screen 2: Web Server Menu Page


PROGESI A PROxy Grammar to Enhance Web Application Firewall for SQL Injection Prevention
DATA OWNER
WEB SERVER
SIGNATURE GENERATOR
SIGNATURE COMPARATOR
END USER
A Malicious SQL Injection Attack
Welcome:: Signature Generator
S-Generator Menu
Image Gallery

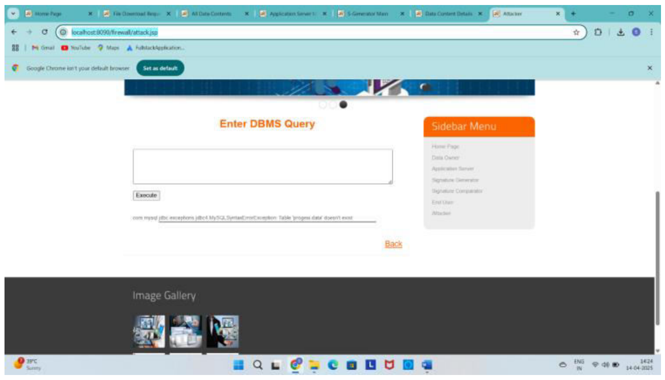
Screen 3 : Signature Generator


PROGESI A PROxy Grammar to Enhance Web Application Firewall for SQL Injection Prevention
DATA OWNER
WEB SERVER
SIGNATURE GENERATOR
SIGNATURE COMPARATOR
END USER
What is SQL Injection Attack?
Welcome to Signature Comparator Main
S-Comparator Menu
Image Gallery

Screen 4: Signature Comparator


PROGESI A PROxy Grammar to Enhance Web Application Firewall for SQL Injection Prevention
DATA OWNER
APPLICATION SERVER
SIGNATURE COMPARATOR
END USER
A Malicious SQL Injection Attack
Hacker injects malicious SQL query via input data
Malicious SQL query is validated and command executed by the database
Hacker is granted access to view and alter records
View All Docs Rank in Chart
(Rank is Based on No.of Time Download)
Server Menu
JS Chart
Back
Image Gallery

Screen 5: Rank Chart



```
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Table 'progesi.data' doesn't exist
```

Screen 6: Sql Injection

VI.CONCLUSION

Successful SQL injection (SQLi) attacks pose a serious threat to the confidentiality, integrity, and availability of data, making their detection and prevention a critical concern in the domain of web application security. Addressing this challenge, the paper introduced PROGESI, a grammar-based proxy framework designed to detect and mitigate SQLi attacks. PROGESI can operate independently or complement existing next-generation firewall (NGFW) solutions without degrading performance, particularly in terms of false positive rates. Experimental evaluations demonstrated the effectiveness of PROGESI, even in environments where security by design principles were already applied, by uncovering and patching complex SQLi vulnerabilities that might otherwise evade detection. Additionally, PROGESI maintained high sensitivity levels, despite not utilizing the maximum possible rate of mutation techniques. As part of future research, further exploration will focus on optimizing the generalization parameter, which governs detection breadth and rule accuracy. Selective filtering of low-impact mutations prior to processing may further reduce the execution time required for patch rule generation, enhancing the framework's efficiency and scalability.

REFERENCES

[1] C. Rohith and R. S. Batth, ''Cyber warfare: Nations cyber conflicts, cyber coldwar between nations and its repercussion,'' in Proc. Int. Conf. Comput. Intell. Knowl. Economy (ICCIKE), Dec. 2019, pp. 640–645.

[2] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, ''From ChatGPT to ThreatGPT: Impact of generative AI in cybersecurity and privacy,'' IEEE Access, vol. 11, pp. 80218–80245, 2023.

[3] K. Neupane, R. Haddad, and L. Chen, ''Next generation firewall for network security: A survey,'' in Proc. SoutheastCon, Apr. 2018, pp. 1–6.

[4] J. Liang and Y. Kim, ''Evolution of firewalls: Toward securer network using next generation firewall,'' in Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2022, pp. 0752–0759.

[5] M. T. Arefin, M. R. Uddin, N. A. Evan, and M. R. Alam, ''Enterprise network: Security enhancement and policy management using next-generation firewall (NGFW),'' in Computer Networks, Big Data and IoT. Springer, 1007, pp. 753–769. [Online]. Available: https://link.springer.com/book/10.1007/978-981- 16-0965-7?source=shoppingads&locale=en-it&gad_source=1&gclid=

[6] C. Togay, A. Kasif, C. Catal, and B. Tekinerdogan, ''A firewall policy anomaly detection framework for reliable network security,'' IEEE Trans. Rel., vol. 71, no. 1, pp. 339–347, Mar. 2022.

[7] D. Bringhenti, L. Seno, and F. Valenza, ''An optimized approach for assisted firewall anomaly resolution,'' IEEE Access, vol. 11, pp. 119693–119710, 2023.

[8] E.-S.-M. El-Alfy, ''A heuristic approach for firewall policy optimization,'' in Proc. 9th Int. Conf. Adv. Commun. Technol., Feb. 2007, pp. 236–248. [Online]. Available: https://ceur-ws.org/Vol-3260/paper17.pdf

[9] A. Coscia, V. Dentamaro, S. Galantucci, A. Maci, and G. Pirlo, ''An innovative two-stage algorithm to optimize firewall rule ordering,'' Comput. Secur., vol. 134, Nov. 2023, Art. no. 103423.

[10] L. Schiff and S. Schmid, ''PRI: Privacy preserving inspection of encrypted network traffic,'' in Proc. IEEE Secur. Privacy Workshops (SPW), May 2016, pp. 296–303.

[11] M. Zain ul Abideen, S. Saleem, and M. Ejaz, ''VPN traffic detection in SSL-protected channel,'' Secur. Commun. Netw., vol. 2019, pp. 1–17, Oct. 2019.

[12] J. Heino, A. Hakkala, and S. Virtanen, ''Study of methods for endpoint aware inspection in a next generation firewall,'' Cybersecurity, vol. 5, no. 1, p. 25, Sep. 2022.

[13] A. Coscia, V. Dentamaro, S. Galantucci, A. Maci, and G. Pirlo, ''YAMME: A YAra-byte-signatures metamorphic mutation engine,'' IEEE Trans. Inf. Forensics Security, vol. 18, pp. 4530–4545, 2023.

[14] H. ElSawy, M. A. Kishk, and M.-S. Alouini, ''Spatial firewalls: Quarantining malware epidemics in large-scale massive wireless networks,'' IEEE Commun. Mag., vol. 58, no. 9, pp. 32–38, Sep. 2020.

[15] A. Rahali and M. A. Akhloufi, ‘‘MalBERTv2: Code aware BERT-based model for malware identification,’’ Big Data Cognit. Comput., vol. 7, no. 2, p. 60, Mar. 2023.

[16] P. L. S. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, ‘‘Machine and deep learning solutions for intrusion detection and prevention in IoTs: A survey,’’ IEEE Access, vol. 10, pp. 121173–121192, 2022.

[17] L. Ashiku and C. Dagli, ‘‘Network intrusion detection system using deep learning,’’ Proc. Comput. Sci., vol. 185, pp. 239–247, Jan. 2021.

[18] A. Coscia, V. Dentamaro, S. Galantucci, A. Maci, and G. Pirlo, ‘‘Automatic decision tree-based NIDPS ruleset generation for DoS/DDoS attacks,’’ J. Inf. Secur. Appl., vol. 82, May 2024, Art. no. 103736.

[19] M. S. Islam, M. A. Uddin, D. M. S. Ahmed, and G. Moazzam, ‘‘Analysis and evaluation of network and application security based on next generation firewall,’’ Int. J. Comput. Digit. Syst., vol. 13, no. 1, pp. 193–202, Jan. 2023.

[20] A. Marchand-Melsom and D. B. N. Mai, ‘‘Automatic repair of OWASP top 10 security vulnerabilities: A survey,’’ in Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops, Jun. 2020, pp. 23–30.

[21] S. Alazmi and D. C. De Leon, ‘‘A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners,’’ IEEE Access, vol. 10, pp. 33200–33219, 2022.

[22] F. Faisal Fadlalla and H. T. Elshoush, ‘‘Input validation vulnerabilities in web applications: Systematic review, classification, and analysis of the current state-of-the-art,’’ IEEE Access, vol. 11, pp. 40128–40161, 2023.

[23] H. Gupta, S. Mondal, S. Ray, B. Giri, R. Majumdar, and V. P. Mishra, ‘‘Impact of SQL injection in database security,’’ in Proc. Int. Conf. Comput. Intell. Knowl. Economy (ICCIKE), Dec. 2019, pp. 296–299.

[24] M. Silva, S. Ribeiro, V. Carvalho, F. Cardoso, and R. L. Gomes, ‘‘Scalable detection of SQL injection in cyber physical systems,’’ in Proc. 12th Latin-Amer. Symp. Dependable Secure Comput., Oct. 2023, pp. 220–225.

[25] A. Razaque, F. Amsaad, M. J. Khan, S. Hariri, S. Chen, C. Siting, and X. Ji, ‘‘Survey: Cybersecurity vulnerabilities, attacks and solutions in the medical domain,’’ IEEE Access, vol. 7, pp. 168774–168797, 2019.