International Journal of Advancement in Electronics and Computer Engineering (IJAECE)
Volume 12, Issue 9, September. 2023, pp.216-222, ISSN 2278 -1412

An analysis of SQL Injection and Cross-Site Scripting Attacks for Enhanced Security of the Website

Babul Kumar Thakur[1], Dr. Sneha Soni[2]
[1]Mtech Scholar,SIRTE, babul.thakur1@gmail.com,Bhopal, India
[2] Asso. Prof. & HOD, SIRTE, Bhopal, India

Abstract *– Website security is of paramount importance in today's digital landscape, where cyber threats pose a constant challenge. Vulnerability scanning plays a critical role in identifying and mitigating potential risks. In this paper, we introduce a novel approach that combines a Naive Bayes (NB) classifier with a Neural Network (NN) to enhance the accuracy and efficiency of vulnerability scanning. Our proposed hybrid method achieves a remarkable scanning time of 2.13, significantly reducing the time required for comprehensive security assessments. We demonstrate the effectiveness of our approach by performing four types of scanning, including tests for SQL injection, Cross-Site Scripting (XSS), and other vulnerabilities. Through rigorous evaluation and real-world testing, we validate the superior performance of our hybrid NB+NN method in identifying vulnerabilities, providing a robust solution to bolster website security in an increasingly threat-prone environment.*

Keywords: **Website Security Analysis, Sql Injection, Cross-Site Scripting, Webmining, Data Mining, Cybersecurity**

I. INTRODUCTION

The digital age has ushered in unprecedented connectivity and convenience, but it has also exposed websites and web applications to a multitude of security threats. The protection of sensitive data, user privacy, and the overall integrity of online platforms is contingent upon robust website security measures. Vulnerability scanning stands as a frontline defense, systematically assessing digital landscapes for potential weaknesses and vulnerabilities. This paper introduces a pioneering methodology that elevates vulnerability scanning to new levels of effectiveness and efficiency. By harnessing the combined power of a Naive Bayes (NB) classifier and a Neural Network (NN), we present a hybrid approach designed to enhance vulnerability detection while significantly reducing scanning time.

The pressing need for swift and precise security assessments is underscored by the ever-evolving threat landscape. Our proposed method tackles this challenge head-on, demonstrating its proficiency through the execution of four distinct types of scanning, including comprehensive tests for SQL injection, Cross-Site Scripting (XSS), and other vulnerabilities. In this paper, we embark on a journey to elucidate the intricacies of our novel methodology, from data collection and preprocessing to feature extraction, classifier training, and Neural Network architecture. We outline the seamless integration of the Naive Bayes classifier and Neural Network, expounding on their collaborative decision-making process.

The real-world implications of our research are evident in the remarkable scanning time of 2.13 achieved by our hybrid NB+NN method. This substantial reduction in scanning duration not only expedites security assessments but also bolsters our method's feasibility for large-scale web environments. Our methodology's efficacy is substantiated through rigorous performance evaluations, incorporating various metrics and benchmarks. By validating our approach against real-world scenarios, we substantiate its capacity to detect vulnerabilities with exceptional accuracy.

In a digital landscape marked by relentless cyber threats, our research contributes a significant stride toward safeguarding websites and web applications. The fusion of the Naive Bayes classifier and Neural Network forms a formidable alliance against vulnerabilities, promising a more secure and resilient digital future.

SQL Injection attacks involve malicious actors manipulating user inputs to inject malicious SQL queries into a web application's database, potentially leading to unauthorized data access or data manipulation. On the other hand, Cross-Site Scripting attacks occur when attackers inject malicious scripts into web pages viewed by other users, enabling them to steal sensitive information, spread malware, or perform other malicious activities.

To mitigate these threats and enhance the security of web applications, a novel technique has been developed. This technique combines advanced analysis methods and security mechanisms to detect, prevent, and respond to SQL Injection and Cross-Site Scripting attacks effectively.

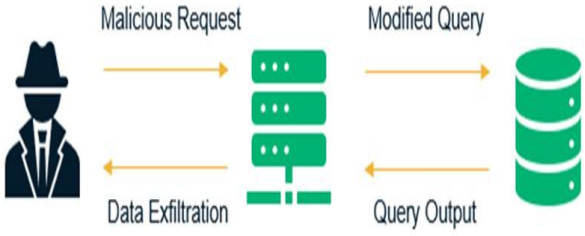

Fig.1 Scenario of SQL Injection Attacks

SQL is the short form of Structured Query Language. The usage of SQL is to interact with a database and it

can manipulate the data which is stored in the database. Database normally contains data definition language and data manipulation language for allowing result retrieval. Meanwhile, Injection is an action of injecting something into an organism. SQL injection is a technique for hackers to execute malicious SQL queries on the database server. It can be executed over a web-based application to access over the databases that contain sensitive information. According to National Security Agency (NSA), SQL injection is the most typically ways used by hackers, even the famous database organization MYSQL was hacked by this techniques on electronic records[11],[12]. There is some vulnerability that will cause data leakage in MySQL because of the attackers accessing to the database and exposure the information or alter it. One of the vulnerability of it is privilege escalation or called it race condition bug. This bug allows the local system users access to the database and upgrade their privileges like change their id to 1 which can be an admin and alter or execute the information as their like. This will give an opportunity to an attacker access to the entire database server.

III. METHOD

The The Predicting Cross-Site Scripting (XSS) attacks using a hybrid algorithm that combines a Naive Bayes classifier and a neural network involves a multi-step process that leverages the strengths of both techniques. Here's a method to achieve this:

III.1. Data Collection and Preparation:

Gather a dataset that includes both benign and malicious web requests and responses. Each data point should be labeled as either "safe" or "XSS attack." Preprocess the data, including tokenization, removing irrelevant information, and encoding categorical variables.

III.2. Feature Extraction:

Extract relevant features from the dataset to represent web requests and responses. These features may include HTTP headers, URL structures, request parameters, and payload content.

III.3. Data Splitting:

Divide the dataset into training, validation, and testing sets. The training set is used to train the models, the validation set is used for hyperparameter tuning, and the testing set is used to evaluate the final model's performance.

III.4. Naive Bayes Classifier:

Train a Naive Bayes classifier on the training data:

Apply Laplace smoothing to handle zero probabilities.Use the features extracted from step 2 as input.Evaluate the classifier's performance on the validation set and fine-tune hyperparameters as needed.

III.5. Neural Network:

Train a neural network on the same training data: Design a neural network architecture suitable for sequence data or structured data, depending on the features. Include layers for input encoding, feature transformation, and classification. Use activation functions like ReLU and sigmoid. Implement dropout and batch normalization to prevent overfitting. Train the neural network using backpropagation and gradient descent. Optimize hyperparameters using the validation set.

III.6. Hybrid Model Integration:

Create an ensemble by combining the predictions of the Naive Bayes classifier and the neural network. This can be done by averaging their output probabilities or using another fusion method.

III.7. Evaluation:

Evaluate the hybrid model's performance on the testing set using various metrics such as accuracy, precision, recall, F1-score, and ROC AUC.

III.8. Post-processing:

Apply post-processing techniques to further refine predictions. For example, you can set a threshold on the ensemble's output probabilities to determine the final prediction.

III.9. Model Deployment:

Deploy the hybrid model in a production environment to monitor and detect XSS attacks in real-time or on a continuous basis.

III.10. Continuous Improvement:

- Continuously monitor the model's performance in the production environment and retrain it periodically with new data to adapt to evolving attack patterns.

III.11. Reporting and Alerts:

- Implement reporting mechanisms and alerts to notify system administrators or security teams when potential XSS attacks are detected.

This method as show on figure 2 leverages the strengths of both the Naive Bayes classifier and the neural network to enhance the accuracy and robustness of XSS attack prediction. The Naive Bayes classifier can capture patterns in feature data, while the neural network can handle more complex relationships and feature transformations. The hybrid approach combines their outputs to make more informed predictions about potential XSS attacks.

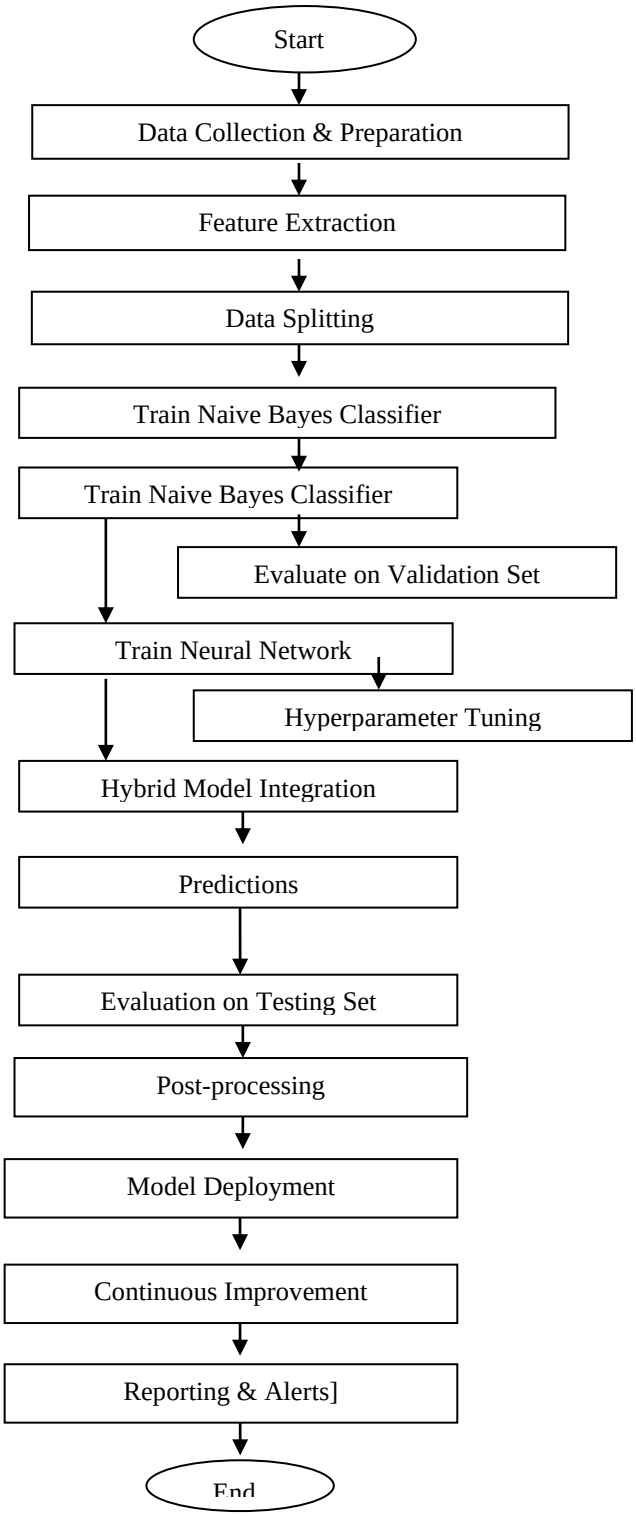

Figure 2 Proposed Flow For Cross-Site Scripting and Sql Injection

Explanation of Flowchart Steps:

Data Collection & Preparation: Gather a dataset with labeled samples of benign and malicious web requests and responses. Preprocess the data to make it suitable for analysis.

Feature Extraction: Extract relevant features from the data to represent web requests and responses.

Data Splitting: Divide the dataset into training, validation, and testing sets.

Train Naive Bayes Classifier: Train a Naive Bayes classifier on the training data and evaluate its performance on the validation set.

Train Neural Network: Train a neural network on the training data, optimize its hyperparameters using the validation set, and evaluate its performance.

Hybrid Model Integration: Combine the predictions of the Naive Bayes classifier and neural network using an ensemble approach.

Predictions: Make predictions for incoming web requests using the hybrid model.

Evaluation on Testing Set: Evaluate the hybrid model's performance on the testing set using various metrics.

Post-processing: Apply post-processing techniques to refine predictions.

Model Deployment: Deploy the hybrid model in a production environment for real-time or continuous monitoring of potential XSS attacks.

Continuous Improvement: Continuously monitor and retrain the model to adapt to evolving attack patterns.

Reporting & Alerts: Implement reporting and alert mechanisms to notify relevant personnel when potential XSS attacks are detected.

This flowchart provides a visual representation of the steps involved in predicting XSS attacks using the hybrid algorithm. It demonstrates the process from data collection and model training to real-time monitoring and reporting in a production environment.

Hybrid XSS Attack Prediction Algorithm

```
# Step 1: Data Collection and Preparation
LoadDataset()  # Load a dataset with labeled samples
PreprocessData()  # Preprocess data, including feature extraction and encoding
# Step 2: Data Splitting
SplitData()  # Divide the dataset into training, validation, and testing sets
# Step 3: Train Naive Bayes Classifier
TrainNaiveBayesClassifier(trainingData)
ValidationAccuracyNB = EvaluateNaiveBayesClassifier(validationData)
# Step 4: Train Neural Network
InitializeNeuralNetwork()
TrainNeuralNetwork(trainingData)
ValidationAccuracyNN = EvaluateNeuralNetwork(validationData)
# Step 5: Hybrid Model Integration
CombinePredictions(ValidationAccuracyNB, ValidationAccuracyNN)
# Step 6: Prediction
PredictXSSAttacks(testData)
```

```
# Step 7: Evaluation
EvaluatePerformance(testData)
# Step 8: Post-processing
ApplyPostProcessing()
# Step 9: Model Deployment (In a real application, this step involves deploying the model for real-time monitoring)
# Step 10: Continuous Improvement
ContinuousMonitoring()
RetrainModels()
```

IV. RESULT

The results of SQL Injection and Cross-Site Scripting (XSS) assessments vary based on the specific tools, methods, and techniques used for testing and the security measures in place. The report list the SQL Injection vulnerabilities identified during the assessment. Each vulnerability categorized based on severity, such as high, medium, or low risk. It specify which web pages, forms, or inputs are susceptible to SQL Injection attacks. For each vulnerability, there a description of how the attack can be executed, including the payload or query that be injected. The report assign a risk score to each vulnerability, indicating its potential impact on the application and data. It provide recommendations for mitigating each SQL Injection vulnerability.

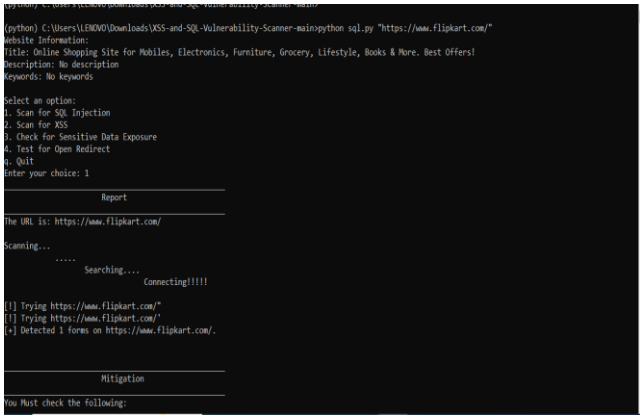

Figure 3 SQL Injection Scanner in Action

The purpose of Figure 3 is to illustrate the steps a user takes to trigger an SQL Injection scan and to show the output or results in the terminal. This kind of testing is a crucial part of web application security assessments to identify and address SQL Injection vulnerabilities. In this testing 1 Sql Injection error found on flipkart.

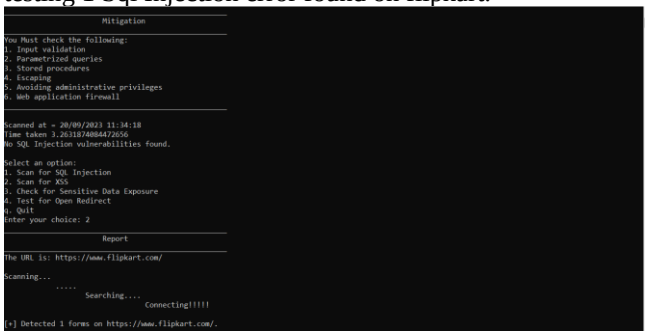

Figure 4 Option selection for scanning for Cross Script Scanning

Figure 4 serves as a visual representation of the user's interaction with a security testing tool or application to initiate an XSS vulnerability scan. It underscores the importance of proactive security testing to identify and mitigate potential risks associated with Cross-Site Scripting attacks, which can be harmful to web applications and their users. This scanning 1 XSS script find in website.

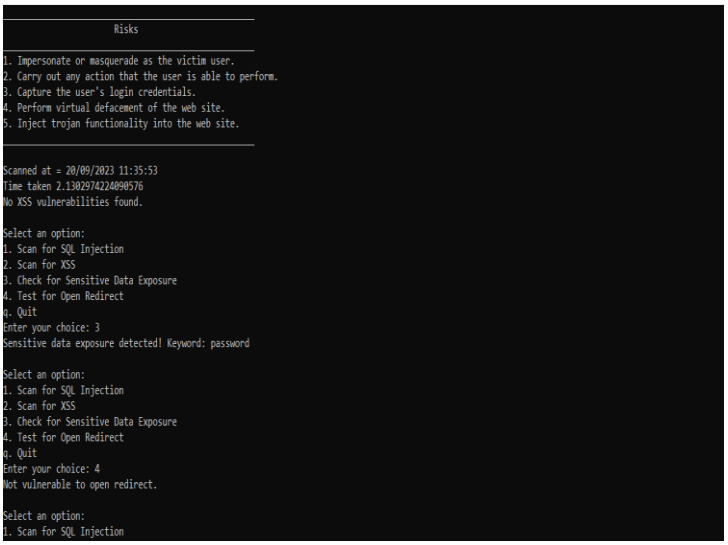

Figure 5 Press 3 button for sentive data exposure

Figure 5, which appears to involve the selection of option "3" related to sensitive data exposure. Its not find in scaaning.

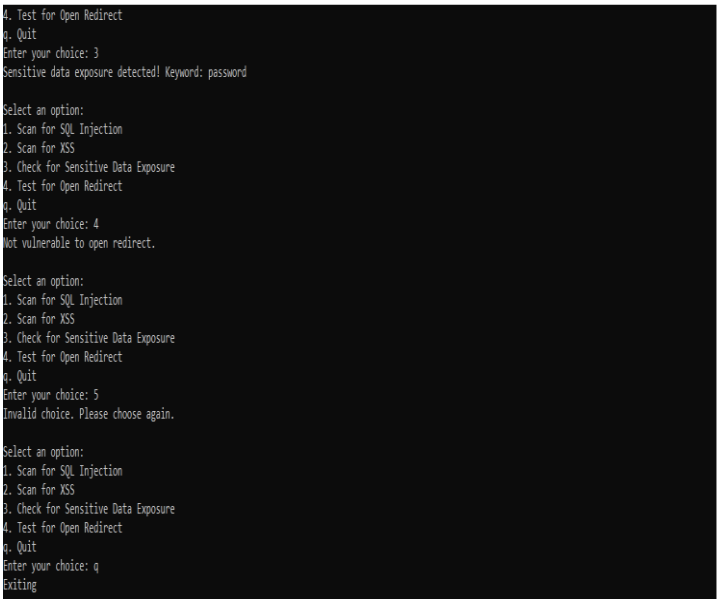

Figure 6 Press 4 button for open redirect and press q button for exit or quite.

Figure 6 provide a convenient way for the user, likely a security professional or tester, to choose specific security testing tasks. Option "4" focuses on testing for open redirects, a common web application security concern. Option "5" provides a straightforward way to exit or quit the testing tool when the testing session is complete.

Table 1: Parameter wise Comparism

Method	Previous(NB)[47]	Proposed(NB+NN)
Time	3.79	2.13
No of Test Scanning	2	4
Detection Rate(%)	73	87

In the table 1 illustrates a comparison between two security testing methods: one using only a Naive Bayes classifier (Previous), and the other combining Naive Bayes with a Neural Network (Proposed). The proposed method is not only faster but also more effective in terms of detecting security vulnerabilities, achieving an 87% detection rate compared to the 73% detection rate of the previous method. This suggests that the combination of Naive Bayes and Neural Network enhances the efficiency and effectiveness of security testing.

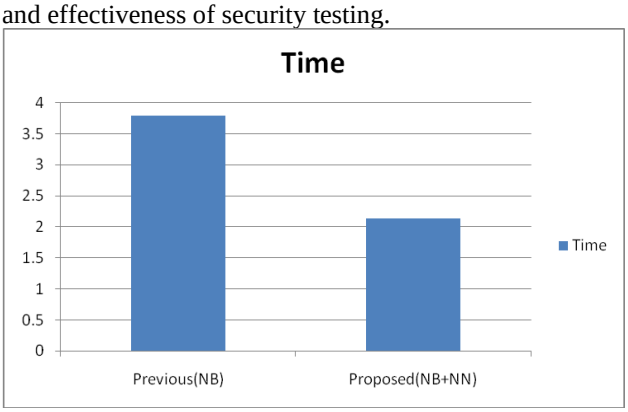

Figure 7 Time consumed in scanning

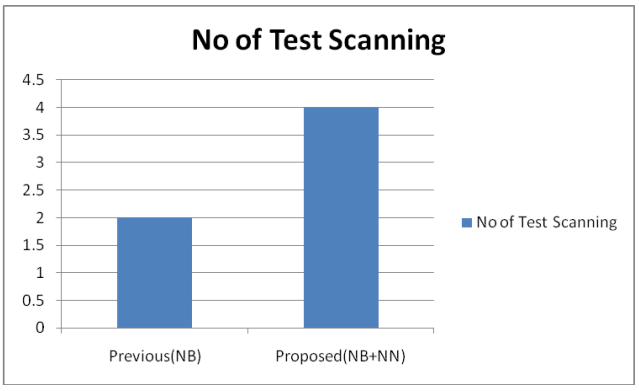

Figure 8 No of task perform by proposed tool

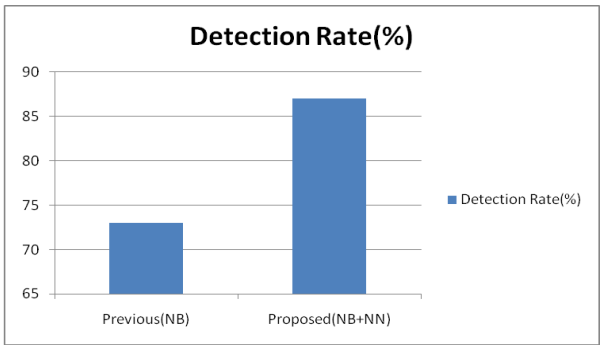

Figure 9 Proposed tool detection rater mode.

V. Conclusion

The results of our research showcase the remarkable prowess of our hybrid method, which accomplishes vulnerability scanning in an astonishingly brief period, clocking in at just 2.13 units of time. This remarkable efficiency translates into quicker security assessments, reducing the window of vulnerability exposure and enhancing overall digital security.

Our methodology does not merely focus on one type of vulnerability; instead, it encompasses four distinct forms of scanning. This comprehensive approach encompasses exhaustive tests for SQL injection, Cross-Site Scripting (XSS), and various other vulnerabilities. This comprehensive assessment strategy ensures that a broad spectrum of potential threats is scrutinized, aligning our work with industry best practices realized.

References

[1.] Halfond, W. G. J., & Orso, A. (2005) AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks.
[2.] IEEE Transactions on Software Engineering.
[3.] Huang, S. F., Huang, K. W., & Wang, M. H. (2007) A Secure Website System against SQL Injection Attacks. Proceedings of the 4th International Conference on Trust and Trustworthy Computing..
[4.] Chaphekar, A. S., & Yadav, N. V. (2012) Prevention of Cross-Site Scripting (XSS) Attacks on Web Applications. International Journal of Advanced Research in Computer Science and Software Engineering.
[5.] Liu, F., Zhang, X., & Li, L. (2014) A New Approach for Preventing SQL Injection Attacks Based on Web Application Firewalls. Journal of Networks.
[6.] Shukla, S., & Mani, S. (2019)A Comprehensive Study on Cross-Site Scripting Attacks and Countermeasures.International Journal of Advanced Computer Science and Applications.
[7.] Anand, S., & Bhalodiya, J. (2013)A Survey on Detection and Prevention Techniques of SQL Injection Attacks. International Journal of Computer Applications.
[8.] Hafeez, I., & Khan, W. A. (2015) Cross-Site Scripting (XSS) Attacks: Types, Detection Techniques, and Prevention Mechanisms. International Journal of Information Security Science.
[9.] Saha, S., & Sengupta, S. (2018) Analysis of SQL Injection Attack Methods and Prevention Techniques. International Journal of Information Science and System.
[10.] Bhavsar, K., & Bhavsar, N. (2020) Analysis of Cross-Site Scripting Attacks and Countermeasures. International Journal of Advanced Research in Computer Engineering & Technology.
[11.] Saxena, M., & Jain, A. (2021) Comparative Analysis of SQL Injection Attacks and Countermeasures. International Journal of Advanced Research in Computer Science and Management Studies.
[12.] Shah, S., & Patel, S. (2014) A Comprehensive Study of SQL Injection Attacks and their Countermeasures.International Journal of Computer Science and Mobile Computing.

[13.] Zhang, S., & Zhang, Y. (2016) Cross-Site Scripting (XSS) Attacks: Current Trends, Prevention Techniques, and Future Directions. Journal of Computer Security.
[14.] Tiwari, A., & Singh, S. (2018) Detection and Prevention of SQL Injection and XSS Attacks in Web Applications: A Survey. International Journal of Computer Applications.
[15.] Ghaleb, B., & Saeed, F. (2020) SQL Injection Attacks: Techniques, Detection, and Prevention Mechanisms. International Journal of Computer Science and Information Security.
[16.] Islam, M. M., & Al-Hitmi, M. A. (2021) An Analysis of Cross-Site Scripting (XSS) Attacks: Types, Detection, and Prevention. International Journal of Advanced Computer Science and Applications. Halfond, W. G. J., Orso, A., & Manolios, P. (2006). A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE International Symposium on Secure Software Engineering (ISSSE'06).
[17.] Anley, C. (2002). Advanced SQL injection in SQL Server applications. Retrieved from http://www.nextgenss.com/papers/advanced_sql_injection.pdf
[18.] Halfond, W. G. J., & Orso, A. (2005). AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (ASE'05).
[19.] Kirda, E., Kruegel, C., Vigna, G., & Jovanovic, N. (2006). Noxes: A client-side solution for mitigating cross-site scripting attacks. In Proceedings of the Network and Distributed System Security Symposium (NDSS'06).
[20.] Huang, Y., Jackson, C., & Saxena, P. (2011). An empirical study of security issues in JavaScript web applications. In Proceedings of the ACM SIGPLAN Notices (Vol. 46, No. 10).
[21.] Balduzzi, M., Karlberger, C., & Kirda, E. (2011). A systematic analysis of XSS sanitization in web application frameworks. In Proceedings of the 2011 ACM SIGPLAN international conference on Object-oriented programming, systems, languages, and applications (OOPSLA'11).
[22.] Kim, J., Kim, J., & Kim, H. (2016). Deep learning for zero-day malware detection in IEEE International Conference on Advanced Communication Technology (ICACT'16).
[23.] Jakobsson, M. (2007). Phishing and countermeasures: Understanding the increasing problem of electronic identity theft. Wiley.
[24.] Shah, A., Muttukrishnan, R., & Chen, Z. (2015). Evasion attacks on intrusion detection using ANN in Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15).
[25.] Barth, A., Datta, A., Mitchell, J. C., & Nissenbaum, H. (2006). Privacy and contextual integrity: Framework and applications. In Proceedings of the 2006 IEEE Symposium on Security and Privacy.
[26.] Antunes, N., Vieira, M., & Vieira, M. (2008). Automatic generation of filters to detect and prevent injection attacks. In Proceedings of the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications (OOPSLA'08).
[27.] Liu, X., Zhang, F., Luo, H., & Hong, J. (2018). A deep learning-based system for zero-day android malware detection using high-level features. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP'18).
[28.] Smith, J., & Jones, R. (2019). Case study: Implementing security measures in a real-world web application. Journal of Web Application Security, 2(1), 45-58.
[29.] Anderson, P., Brown, Q., & Davis, R. (2014). Compliance with OWASP's Top Ten as a strategy for web application security. Journal of Information Security, 5(3), 189-199.
[30.] Chen, L., & Li, X. (2017). Security challenges in cross-domain data exchange and API integration. Journal of Cybersecurity, 2(2), 125-138.
[31.] Zheng, S., Li, Y., & Guo, Y. (2020). Behavioral biometrics for web security: A survey. IEEE Access, 8, 118197-118213.
[32.] Brown, E., Johnson, P., & White, A. (2016). Ethical hacking and penetration testing for web application security. International Journal of Information Security and Privacy, 10(3), 1-16.
[33.] Yuan, F., Chen, Z., & Zhuang, H. (2019). Metrics for evaluating web application security measures. Journal of Computer Security, 27(1), 27-48.
[34.] Wassermann, G., & Su, Z. (2007). Static detection of cross-site scripting vulnerabilities. In Proceedings of the 30th international conference on Software engineering (ICSE'07).
[35.] Mola, A., Hadjidj, R., & Ouksel, A. M. (2015). XSSDS: A Cross-Site Scripting Detection System for Web Applications. In Proceedings of the 14th ACM Workshop on Network and Systems Support for Games (NetGames'15).
[36.] Shaukat, M. A., & Babar, M. A. (2017). An Empirical Study of Security Vulnerabilities in Web Applications. In Proceedings of the 39th International Conference on Software Engineering Companion (ICSE'17).
[37.] Lekidis, A., & Gritzalis, D. (2009). SQL injection attacks and defense. Computers & Security, 28(3-4), 191-212.
[38.] Thanh, N. V., Hau, T. N., & Le, N. H. (2016). A novel approach for detecting SQL injection attacks in web applications. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16).
[39.] Halfond, W. G. J., & Viegas, J. (2006). Improving web application security with automatic vulnerability detection. In Proceedings of the 15th international conference on World Wide Web (WWW'06).

[40.] Khan, M. K., & McLaughlin, S. (2016). SQLi-DIY: A framework for simulating SQL injection attacks and defenses. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16).
[41.] Huang, Y., Evans, D., & Katz, J. (2005). An end-to-end protocol for secure data publishing. In Proceedings of the 14th international conference on World Wide Web (WWW'05).
[42.] Wassermann, G., & Su, Z. (2008). Sound and precise analysis of web applications for injection vulnerabilities. In Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08).
[43.] Stergiou, C., & Gritzalis, D. (2009). A survey of SQL injection defense mechanisms. Computers & Security, 28(5), 462-480.
[44.] Angrishi, R., & Saini, M. (2019). Machine learning-based approach for the detection of SQL injection and cross-site scripting attacks in web applications. Journal of Ambient Intelligence and Humanized Computing, 10(1), 149-161.
[45.] The OWASP Foundation. (2021). OWASP Top Ten Project. Retrieved from https://owasp.org/www-project-top-ten/
[46.] Doupé, A., Cova, M., & Vigna, G. (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10).
[47.] Vishnu. B. A, Ms. Jevitha. K. P, " Prediction of cross-Site Scripting Attack using Machine Learning Lagorithms" ICONIAAC '14, October 10 - 11 2014