Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 16, Issue. 1: 2025
ISSN : **1906-9685**

Web Security Threat Detection

Prof.B. AMARNATHREDDY[1], KUNCHALA HEMANTH[2]
#1 Assistant Professor #2 M.C.A Scholar
Department of Master of Computer Applications,
Qis College of Engineering and Technology

ABSTRACT:

Web applications play a crucial role in modern society, offering a wide range of services from e-commerce to social networking. However, they are also a common target for cyberattacks due to their complexity and the vast amount of sensitive information they handle. Web vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF), pose significant threats to the security of web applications and their users.This paper provides an overview of common web vulnerabilities, their impact, and current approaches for mitigating them. We discuss the importance of secure coding practices, regular security audits, and the use of security tools and frameworks to protect web applications from potential attacks. Additionally, we explore emerging trends and technologies, such as machine learning and artificial intelligence, that show promise in improving web application security.By understanding the nature of web vulnerabilities and implementing appropriate security measures, developers and organizations can enhance the security posture of their web applications and protect against potential threats.

INTRODUCTION:

Web applications play a crucial role in modern society, facilitating various online activities such as e-commerce, social networking, and information sharing. However, the widespread use of web applications also makes them a target for malicious actors seeking to exploit vulnerabilities for nefarious purposes. Web vulnerabilities can lead to data breaches, financial loss, and damage to an organization's reputation.Web vulnerability refers to a weakness or flaw in a web application that can be exploited by attackers to compromise the security of the application or the data it processes. These vulnerabilities can exist at various levels of the web application stack, including the web server, application server, database server, and client-side scripts.Common web vulnerabilities include SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and insecure direct object references. These vulnerabilities can be exploited to steal sensitive information, modify data, or execute malicious code on the user's device.In recent years, the number and complexity of web vulnerabilities have increased, driven by the growing sophistication of attacks and the rapid evolution of web technologies. As a result, organizations must take proactive measures to identify and mitigate web vulnerabilities to protect their assets and maintain the trust of their users. The proposed system for managing web vulnerabilities aims to address the limitations of the existing system by leveraging advanced technologies and approaches. One key aspect of the proposed system is the integration of automated vulnerability scanning tools with machine learning algorithms.By using machine learning, the system can improve the accuracy of

vulnerability detection and reduce false positives and false negatives. Machine learning models can be trained on large datasets of known vulnerabilities to recognize patterns and anomalies in web application code, making them more effective at identifying potential vulnerabilities.Another key component of the proposed system is the use of continuous security testing and monitoring. Rather than relying on periodic scans or manual reviews, the system continuously monitors web applications for vulnerabilities and alerts developers in real-time. This proactive approach allows vulnerabilities to be identified and addressed promptly, reducing the risk of exploitation.Additionally, the proposed system emphasizes the importance of secure coding practices and developer training. Developers are provided with tools and resources to help them write secure code, such as secure coding guidelines and automated code analysis tools. Regular training sessions and workshops are also conducted to raise awareness about web vulnerabilities and best practices for mitigating them.

LITERATURE REVIEW:

1. **Huang et al. (2003) – "Web Application Security Assessment by Fault Injection and Behavior Monitoring"**

- **Approach:** Dynamic analysis by fault injection to discover vulnerabilities.
- **Merits:**
  - Can detect runtime vulnerabilities.
  - Works without needing access to source code.
- **Demerits:**
  - High false positives.
  - Requires significant computational resources.

2. **Williams and Wichers (2006) – OWASP Top Ten Project**

- **Approach:** Community-driven list identifying the top 10 critical web application security risks.
- **Merits:**
  - Widely accepted and updated regularly.
  - Educates developers and auditors.
- **Demerits:**
  - Descriptive but not a detection method.
  - Not exhaustive; only covers the top 10.

3. **Musch, M. et al. (2018) – "A Survey on Web Application Vulnerability Detection Tools"**

- **Approach:** Comparative study of static, dynamic, and hybrid tools.
- **Merits:**
  - Evaluates real-world tools.
  - Identifies gaps in coverage and performance.
- **Demerits:**
  - Lacks original detection techniques.
  - Results may vary based on test cases.

4. **Fonseca et al. (2007) – "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks"**

- **Approach:** Empirical evaluation of scanning tools like Acunetix, Nessus, etc.
- **Merits:**
  - Practical relevance for developers and testers.
  - Highlights strengths and weaknesses of each tool.

- **Demerits:**
  - Limited to specific attack types (SQLi, XSS).
  - May not reflect newer vulnerabilities.

5. **Antunes & Vieira (2015) – "Comparing the Effectiveness of Penetration Testing and Static Code Analysis"**

- **Approach:** Comparison of static code analysis vs. penetration testing.
- **Merits:**
  - Highlights trade-offs between early detection and real-world simulation.
  - Useful for choosing the right technique.
- **Demerits:**
  - Does not provide new detection algorithms.
  - Static analysis may miss logic flaws.

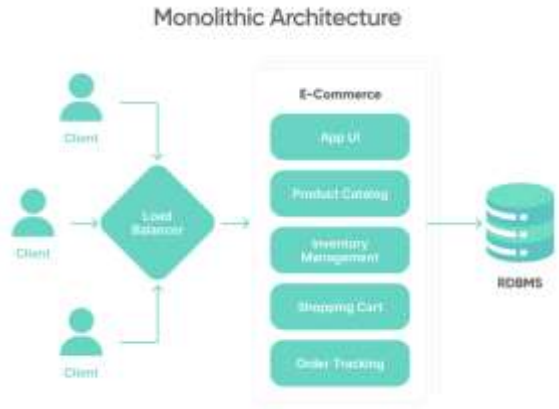
6. **Doupe et al. (2010) – "Wepawet: An Automated Web Exploit Detection System for JavaScript"**

- **Approach:** Automated dynamic analysis of JavaScript code.
- **Merits:**
  - Effective against obfuscated malware.
  - Scalable cloud-based implementation.
- **Demerits:**
  - Focuses only on client-side code.
  - May not detect server-side issues.

7. **Li et al. (2011) – "Parameterized Unit Testing for Web Security"**

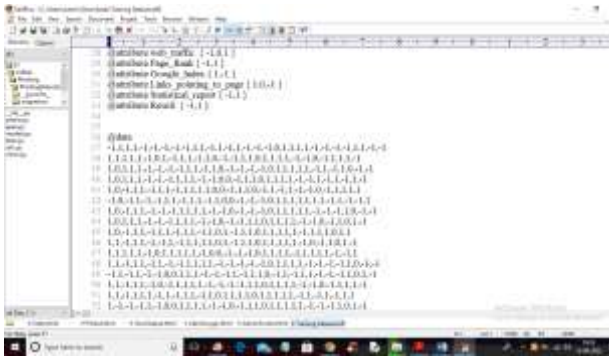
- **Approach:** Uses unit testing with parameters to test web apps.
- **Merits:**
  - Integrates with development workflows.
  - Low false positives.
- **Demerits:**
  - Requires developer expertise.
  - Not effective without thorough test coverage.

SYSTEM ARCHITECTURE:

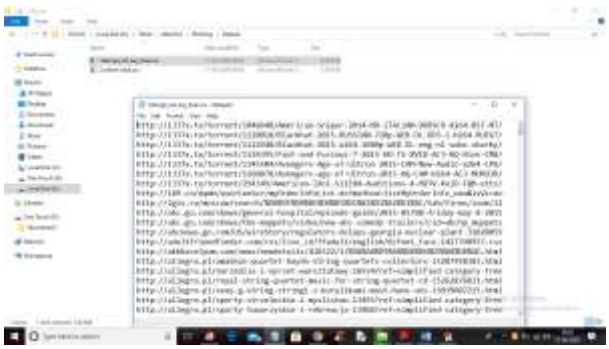


RESULTS:

In this project we are implementing SVM and Light GBM machine learning algorithms o detect phishing website URLS. We are training all this algorithms with normal and phishing URLS and build a trained model and this train model will be applied on new TEST URL to detect whether its normal or phishing URL. In this project you asked to use UCI machine learning phishing dataset but this dataset contains only 0's and 1's values like below screen



From above dataset ML algorithms can get trained but we can't understand anything so I am using REAL WORLD URL dataset which contains normal and phishing URLS like below screen



In above screen you can see our dataset contains 2 folders called benign (phishing URLS) and valid (normal URL) and this are real world URLS and we will train all algorithms with above dataset and then when we input any test URL then ML model will predict as normal or phishing

To run this project double click on 'run.bat' file to start python DJANO server like below screen

In above screen DJANGO webserver started and now open browser and enter URL http://127.0.0.1:8000/index.html and press enter key to get below output

In above screen click on 'Admin Login Here' link to get below login screen

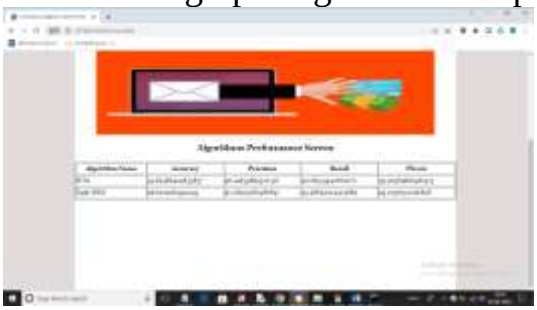
In above screen enter username and password as 'admin' and 'admin' and then press button to get below output

In above screen click on 'Run SVM Algorithm' link to train SVM algorithm and get below output

In above screen we can see SVM confusion matrix where x-axis represents predicted class and y-axis represents TRUE class and we can see SVM predict 2977 records correctly as NORMAL and only 145 are incorrect prediction and it predict 824 records as PHISHING URL and only 26 are incorrect prediction and now close above graph to get below output

In above screen with SVM we got 95% accuracy and now click on 'Run Light GBM Algorithm' link to get below output

In above screen we can see Decision Tree confusion matrix graph and now close above graph to get below output



In above screen with Light GBM also we got 96% accuracy and now click on 'Test Your URL' link to get below screen

In above screen enter any URL and then press button and then Light GBM will predict whether that URL IS normal or phishing

In above screen I entered URL as https://mail.google.com and then press button to get below output

In above screen in blue colour text we can see given URL predicted as GENUINE (normal) and now test other URL. Similarly now I will enter Google.com in below screen

In above screen I gave URL as Google.com and below is the output

In above screen Google.com also predicted as Genuine. Now in below screen from internet I am taking one phishing URL and then input to my application to get prediction

In above screen blue colour URL is the phishing URL and I will input that to my application in below screen and below is the phishing URL from internet

'https://in.xero.com/3LQDhRwfvoQfeDtlDMqkk1JWSqC4CMJt4VVJRsGN'

In above screen I entered same URL and press button to get below output

In above screen in blue colour text we can see application detected PHISHING in given URL and similarly you can enter any URL and detect it as NORMAL or phishing

CONLUSION

In conclusion, web vulnerabilities pose a significant threat to the security and integrity of web applications. Common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) can be exploited by attackers to steal sensitive information, modify data, or execute malicious code. To mitigate these vulnerabilities, organizations should implement secure coding practices, such as input validation and output encoding, to prevent vulnerabilities from being introduced into web applications. Additionally, the use of automated tools for vulnerability detection and regular security audits can help identify and address vulnerabilities in existing web

applications.It is crucial for organizations to stay vigilant and proactive in addressing web vulnerabilities, as the threat landscape continues to evolve. By taking proactive measures to secure their web applications, organizations can reduce the risk of exploitation and protect their assets and reputation**.**

FUTURE WORK:

1. **AI and Machine Learning-Based Detection**

- **Future Direction**: Develop adaptive models that can learn from web traffic patterns and detect zero-day attacks.
- **Justification**: Traditional signature-based methods cannot detect novel or obfuscated vulnerabilities.
- **Challenge**: Reducing false positives and maintaining model accuracy in real-time environments.

2. **Automated Vulnerability Testing Frameworks**

- **Future Direction**: Create fully automated, intelligent frameworks for vulnerability discovery and remediation.
- **Justification**: Manual testing is time-consuming and often incomplete.
- **Challenge**: Automating logical flaw detection and contextual security analysis.

3. **Security for Modern Web Technologies**

- **Future Direction**: Explore vulnerabilities in Single Page Applications (SPAs), Progressive Web Apps (PWAs), and WebAssembly.
- **Justification**: These technologies are increasingly used but less understood in terms of their security implications.
- **Challenge**: Tooling support and lack of mature security best practices.

4. **Vulnerability Detection in APIs and Microservices**

- **Future Direction**: Design security testing tools specifically tailored for REST, GraphQL, and microservice communication.
- **Justification**: Web apps are increasingly backend-driven with APIs as primary attack vectors.
- **Challenge**: Dynamic environments and complex access control policies.

5. **Human-Centric Security Models**

- **Future Direction**: Incorporate user behavior and human error modeling into web security tools.
- **Justification**: Many security breaches stem from social engineering or misconfiguration.
- **Challenge**: Balancing usability and security in real-world applications.

6. **Secure Development Lifecycle Integration**

- **Future Direction**: Embed security testing and vulnerability scanning into CI/CD pipelines.
- **Justification**: Early detection reduces remediation costs and improves quality.
- **Challenge**: Avoiding performance bottlenecks and ensuring developer adoption.

7. **Better Defense Mechanisms Against Client-Side Attacks**

- **Future Direction**: Strengthen defenses against client-side attacks like DOM-based XSS and clickjacking.
- **Justification**: Client-side logic is expanding and becoming a primary target.
- **Challenge**: Browsers vary in support for policies like CSP, and developers often misconfigure them.

8. **Privacy and Compliance-Oriented Security**

- **Future Direction**: Align vulnerability assessment with GDPR, CCPA, and other compliance standards.
- **Justification**: Web vulnerabilities increasingly intersect with data privacy laws.
- **Challenge**: Legal and technical integration of privacy and security testing.

9. **Real-Time Monitoring and Response Systems**

- **Future Direction**: Implement systems for real-time detection and automated mitigation of attacks.
- **Justification**: Rapid response minimizes impact from exploits.
- **Challenge**: Ensuring accuracy and low-latency without affecting performance.

10. **Security Awareness and Developer Education**

- **Future Direction**: Enhance developer tools and training focused on secure coding practices.
- **Justification**: Most vulnerabilities are introduced during development.
- **Challenge**: Keeping training relevant as threats evolve.

REFERENCES:

1. Halfond, W. G., Orso, A., & Manolios, P. (2006). AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'06).

2. Huang, Y., Huang, C., Su, S., & Lee, J. (2014). A Survey on Web Application Security. Journal of Software, 9(1), 218-228.

3. Barth, A., Jackson, C., & Mitchell, J. C. (2008). Robust Defenses for Cross-Site Request Forgery. In Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS '08).

Authors:

Mr. B. Amarnath Reddy is an Assistant Professor in the Department of Master of Computer Applications at QIS College of Engineering and Technology, Ongole, Andhra Pradesh. He earned his M.Tech from Vellore Institute of Technology(VIT), Vellore. His research interests include Machine Learning, Programming Languages. He is committed to advancing research and fostering innovation while mentoring students to excel in both academic and professional pursuits.

Mr. **KUNCHALA HEMANTH** has received his MCA (Masters of Computer Applications) from QIS college of Engineering and Technology Vengamukkapalem(V), Ongole, Prakasam dist., Andhra Pradesh-