International Research Journal of Education and Technology
Peer Reviewed Journal
ISSN 2581-7795

AUTOPENT: AN ADVANCED AUTOMATED WEB VULNERABILITY SCANNER

A Modular Approach for Secure Web Applications

SESHAIAH REPUDI, M. Tech [CSE] at UNIVERSAL COLLEGE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS), Approved by A.I.C.T.E. New Delhi || Affiliated to JNTUK, Accredited with 'B++' Grade by NAAC, Kakinada. Perecharla (Narasaraopet Road), Dokiparru (Post), Medikonduru (M), Guntur (Dist.), Andhra Pradesh – 522438.
Promoted: The Diocese of Guntur Society, Guntur. https://ucet.edu.in/

G KUSUMA HARINADH, M. Tech (CSE), Asst. Professor, Department of Computer Science & Engineering. Perecharla (Narasaraopet Road), Dokiparru (Post), Medikonduru (M), Guntur (Dist.), Andhra Pradesh – 522438.

Abstract:This research focuses on the increasing reliance on digital technologies, highlighting the critical need for web application security due to the emergence of vulnerabilities that attract malicious actors. AutoPent is an automated, modular web vulnerability scanner created to tackle this issue effectively. It proficiently identifies common vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Remote Code Execution (RCE), Local File Inclusion (LFI), and Open Redirect vulnerabilities. By utilizing advanced web crawling methods, dynamic payload libraries, multi-threaded processing, and robust encoding detection techniques, AutoPent delivers precise and actionable security insights. This paper elaborates on the architecture, methodologies, implementation, and outcomes associated with AutoPent, underscoring its significant contributions to web application security. The tool's user-oriented design, thorough reporting capabilities, and ethical usage guidelines establish it as an essential resource for developers and organizations aiming to bolster their cybersecurity measures.

Index Terms **- Web Security,Penetration Testing,SQL Injection,Automation in Cybersecurity.**

I. INTRODUCTION

Web applications represent essential infrastructure for both businesses and individuals, supporting communication, transactions, and data management. However, the rising sophistication of cyberattacks has revealed considerable weaknesses in conventional security practices. Frequent threats such as SQL Injection, XSS, and RCE pose risks to the confidentiality, integrity, and availability of sensitive information.

Penetration testing, a vital method for evaluating web security, often encounters limitations related to time, cost, and resource availability. Manual testing methods are labor-intensive and susceptible to errors, while many available automated tools concentrate on a limited range of vulnerabilities. AutoPent meets these challenges by providing a scalable, automated solution that combines dynamic web crawling, multi-threaded vulnerability testing, and comprehensive report generation.

This paper offers an in-depth examination of AutoPent's capabilities and technical architecture. The tool's development was influenced by a comprehensive understanding of cybersecurity problems, prioritizing accuracy, efficiency, and user-friendliness. By empowering organizations to proactively discover and remediate vulnerabilities, AutoPent plays a pivotal role in fostering a more secure digital landscape.

II. LITERATURE SURVEY

The development of AutoPent was inspired by existing research and advancements in automated vulnerability detection. Key studies include:

- **Rahul Maini et al. (2020)** developed an automated scanner focused on SQL Injection and XSS vulnerabilities. Their approach combined black-box testing with a web crawler. However, it lacked extensibility for other vulnerability types like RCE or Open Redirects.
- **Mahmoud Mohammadi et al. (2018)** emphasized unit testing for XSS, enabling early detection during development. While effective for specific vulnerabilities, it required integration into the development lifecycle and could not handle post-deployment testing.
- **Alde Alandaa et al. (2017)** demonstrated SQL Injection testing using black-box techniques, revealing alarming vulnerability rates in public web applications. However, their method lacked automation for large-scale application testing.

(1) Existing System

Current methods for securing web applications have seen significant advancements, yet they come with their own set of limitations. One example, found in "Automated Web Vulnerability Scanner" by Rahul Maini, Rahul Pandey, Rajeev Kumar, and Rajat Gupta, employs a black-box testing technique that integrates a port scanner and web crawler. This system detects vulnerabilities such as SQL Injection, XSS, CRLF Injection, and Open Redirects by simulating attacks to evaluate a web application's vulnerabilities while generating detailed reports on its findings. Although it proves effective for enterprise-level applications, its efficiency is hindered by its limited focus on a select few common vulnerabilities.

Another solution, detailed in "Automatic Web Security Unit Testing: XSS Vulnerability Detection" by Mahmoud Mohammadi and team, incorporates security testing into the software development lifecycle. This approach concentrates on XSS vulnerabilities and utilizes automated testing of encoding functions to sanitize inputs. The system creates XSS attack strings to assess input sanitization and is effective in identifying zero-day XSS threats. However, its effectiveness is mainly confined to detecting XSS vulnerabilities, requiring proper integration into development settings for optimal efficiency.

Additionally, the research by Alde Alandaa and colleagues in "Web Application Penetration Testing Using SQL Injection Attack" highlights SQL Injection testing as a fundamental component of penetration testing. This black-box testing method is crucial for uncovering vulnerabilities that could lead to unauthorized database access. Their analysis across various sectors indicates that SQL Injection remains worryingly widespread. Nevertheless, the system's concentration on SQL Injection neglects other vital vulnerabilities, leaving applications vulnerable to a wider range of attacks.

Moreover, the study "Web Application Vulnerabilities and Best Practices" by Pradeep B. Tarakar and Dr. Srinivasan V provides an in-depth examination of common vulnerabilities while presenting best practices such as secure coding and input validation to mitigate risks like CSRF and IDOR. Although these guidelines are essential for enhancing application security, relying exclusively on static best practices does not ensure protection against evolving and complex cyber threats. Furthermore, these methodologies often lack the automation necessary to effectively address the demands of large-scale and rapidly evolving web environments.

(2) Limitations in Existing Methods

Despite significant advancements, current tools and methods face several challenges:

Limitations of Black-Box Testing:

- Black-box scanners replicate known attacks.
- They rely on pre-existing vulnerability databases.
- This method is inadequate against new or advanced threats not listed in the databases.
- The predefined attack libraries are inflexible.
- This inflexibility limits adaptation to rapidly changing threats.
- As a result, applications remain vulnerable to emerging threats.

Narrow Focus on Specific Vulnerabilities:

- Mohammadi et al. highlight the primary focus of security testing in software development on detecting XSS vulnerabilities.
- This narrow emphasis neglects broader threats, including SQL Injection and Remote Code Execution (RCE).
- As a result, applications may only be partially protected against various security threats.
- Developers with limited security expertise may face substantial difficulties during the implementation of security measures.
- Extensive customization may be required, which can impede overall efficiency in the development process.

Restricted Vulnerability Coverage:

- Research by Alandaa et al. focuses solely on SQL Injection vulnerabilities.
- Other important vulnerabilities such as XSS, LFI, and CSRF are not addressed.
- This narrow focus may lead to a false sense of security among users and developers.
- Undetected vulnerabilities can still pose a significant risk and be exploited.
- The complexity of modern web frameworks requires a more comprehensive vulnerability detection system.
- A more adaptable approach to vulnerability detection is essential for effective security.

Dependence on Manual Best Practices:

- Best practices suggested by Tarakar and Dr. Srinivasan V emphasize consistent adherence by developers.
- Manual enforcement of security measures is crucial for effectiveness.
- Human errors and inconsistencies can undermine static security protocols.
- Static practices are less effective against sophisticated and rapidly changing attack strategies.
- There is a growing need for dynamic and automated security solutions.

AutoPent effectively addresses these issues with a holistic, automated, and user-friendly approach.

III. SYSTEM ANALYSIS AND REQUIREMENTS

(1) Problem Statement

The current digital landscape is experiencing a rapid increase in web application development. While these applications provide greater accessibility and convenience, they also become prime targets for cyber threats. Security weaknesses like SQL Injection, XSS, RCE, and LFI can compromise sensitive information and disrupt services, resulting in considerable financial repercussions and damage to reputation. The intricate nature of these applications often exceeds the capabilities of manual testing methods, which tend to be labor-intensive, require substantial resources, and are prone to human errors. Moreover, current automated tools frequently fall short in providing comprehensive protection, as they either concentrate on a limited set of vulnerabilities or fail to keep pace with emerging attack strategies.

(2)Proposed System

The proposed solution, called AutoPent, is intended to serve as an advanced automated web vulnerability scanner that substantially improves security coverage and efficiency. Unlike current solutions that typically concentrate on a narrow range of

vulnerabilities, AutoPent employs a thorough approach for identifying SQL Injection, Cross-Site Scripting (XSS), Remote Code Execution (RCE), Local File Inclusion (LFI), and Open Redirect weaknesses. By utilizing a blend of modern web crawling techniques and multi-threaded execution, the system guarantees efficient scanning of internal URLs, making it apt for both small-scale initiatives and large enterprise applications.

A key feature of AutoPent is its broad utilization of constantly updated payload libraries, which adapt to new attack patterns. This dynamic integration of payloads aids in detecting both prevalent and emerging threats, thus addressing significant shortcomings found in current tools. Furthermore, the scanner utilizes advanced encoding detection through chardet, ensuring that various types of content are accurately parsed and examined. This capability helps preserve the scanner's effectiveness across various web environments, from straightforward HTML-based sites to intricate, JavaScript-heavy web applications.

The system also emphasizes automation and ease of use. It includes an efficient PDF report generation feature that provides detailed summaries and analyses of the identified vulnerabilities. Each report presents a clear overview of affected URLs, details the type and severity of each vulnerability, and offers practical recommendations for remediation. This capability removes the need for manual analysis, delivering users a professional and comprehensible summary of their web application's security status.

Ethical considerations and responsible usage are fundamental aspects of AutoPent's design. The system incorporates features to mitigate misuse, including explicit guidelines for ethical scanning and responsible disclosure. By integrating a robust error-handling system and comprehensive logging, the tool guarantees transparent reporting of any issues that arise during the scanning process. This comprehensive approach fosters safe and responsible security practices while providing users a powerful means to identify and address potential risks efficiently.

(3) Advantages

AutoPent provides several benefits over traditional and automated methods:

Comprehensive Vulnerability Coverage: Allows simultaneous evaluation for SQL Injection, XSS, RCE, LFI, and Open Redirect vulnerabilities.
Scalability: Utilizes multi-threaded execution to maintain efficient operations, even for applications containing thousands of URLs.
Dynamic Adaptability: Regularly updates payload libraries to remain effective against shifting attack vectors.
User-Centric Design: Features an easy-to-navigate interface and detailed PDF reporting, making the tool user-friendly for individuals with varied technical backgrounds.

(4) Functional Requirements

AutoPent is built to include the following functionalities:

1. **Web Crawling and URL Collection: The** system should implement a robust web crawler designed to traverse and gather all internal URLs from the designated web application. It must handle intricate web configurations, like JavaScript-driven navigation, guaranteeing that no page or essential link is overlooked during the exploration process.
2. **Vulnerability Detection:** The scanner must automatically identify and categorize various vulnerabilities, including SQL Injection, XSS, RCE, LFI, and Open Redirects. Each type of vulnerability should be assessed using an extensive array of payloads to enhance detection precision and mitigate potential risks.
3. **Payload Customization:** The system should enable users to tailor payloads for various vulnerability assessments. This functionality is vital for specifically targeting vulnerabilities based on the distinct features of the web application being evaluated, providing increased adaptability in security evaluations.
4. **Automated Report Generation:** At the conclusion of each scan, a PDF report must be produced, summarizing the identified vulnerabilities. This report should classify issues according to severity and include detailed explanations, along with practical recommendations for mitigating each vulnerability, facilitating easy understanding for developers.
5. **Logging and Error Handling**: The system must keep an extensive log of all activities performed, errors encountered, and vulnerabilities identified during the scanning procedure. Error handling should be effective, ensuring the scanner operates seamlessly even in the face of unexpected problems, with minimal impact on overall performance.
6. **User Authentication and Access Control:** The system should have integrated user authentication to limit access to authorized users only. Access control protocols should guarantee that only individuals with the appropriate permissions can initiate scans or access detailed reports, bolstering the tool's security.
7. **Scheduled Scans and Automation:** Users should have the capability to schedule scans to run automatically at predetermined intervals. This functionality is essential for ongoing security oversight, enabling organizations to proactively address potential threats without manual involvement and ensuring timely identification of vulnerabilities.
8. **Multi-Language Support:** The system should be capable of scanning web applications in various languages. This feature ensures that content is effectively parsed and analyzed, irrespective of the language utilized, which is essential for global applications or websites offering multilingual options.

(5) Non-Functional Requirements

1. **Performance and Efficiency:** The system must operate with optimal efficiency, capable of scanning large and intricate web applications within an acceptable timeframe. It should employ multi-threaded execution to enhance speed and resource usage, guaranteeing high performance while maintaining thoroughness.
2. **Scalability:** The system needs to be scalable, accommodating web applications of various sizes, ranging from small blogs to extensive enterprise platforms with thousands of pages. Its architecture should facilitate straightforward scaling, ensuring that performance remains consistent even as the number of URLs or the complexity of the application increases.
3. **Reliability and Stability:** The scanner must provide dependable and consistent results, even under significant load or in demanding web environments. The system should maintain stability and operate effectively without crashing or losing data, delivering reliable security assessments.
4. **Accuracy and Precision:** The mechanisms for detecting vulnerabilities should be extremely accurate, reducing both false positives and false negatives. The system should implement advanced methods, such as dynamic payload analysis and encoding detection, to ensure precise identification of security vulnerabilities, thereby enhancing the credibility of the findings.
5. **Usability and User Experience**: The interface must be user-friendly and easy to navigate, catering to individuals with limited knowledge of cybersecurity. The system should include clear documentation and guided workflows to assist users in configuring scans and interpreting results effectively, making it accessible to a wide audience.
6. **Security and Data Protection:** The system must prioritize the protection of user data and the activities involved in scanning. All information, including logs and reports, should be securely stored and safeguarded against unauthorized access, ensuring the confidentiality of sensitive information and preventing the tool from being misused.
7. **Compliance and Ethical Use:** The tool should be designed with compliance in mind, aligning with industry standards and legal regulations for security testing. It should integrate ethical use features, such as acceptance of terms of service and warnings regarding unauthorized scanning, to deter misuse and inform users about legal constraints.
8. **Cross-Platform Compatibility:** The system should function seamlessly across various operating systems, including Windows, macOS, and Linux. This compatibility ensures that a diverse set of users can utilize the tool on their preferred platforms without encountering compatibility issues.
9. **Resource Optimization:** The scanner should be optimized for efficient use of system resources, preventing unnecessary consumption of CPU, memory, or network bandwidth. This optimization is particularly important in shared server environments or scenarios with limited resources to minimize the impact on other operations.
10. **Extensibility:** The architecture of the system should be modular, allowing for easy future enhancements and updates. Incorporating new features, such as additional vulnerability checks or integrations with threat intelligence platforms, should be straightforward, ensuring the tool remains relevant and up-to-date with changing security requirements.

(6) Feasibility Study

The AutoPent Web Vulnerability Scanner is an advanced automated tool created to bolster the security of web applications. This feasibility study examines the technical, financial, and operational viability of the system, while also identifying potential risks and obstacles.

Technical Feasibility

The technical feasibility of AutoPent is high, as it is built upon established technologies and libraries, including requests, BeautifulSoup, chardet, FPDF, and multi-threading techniques. Utilizing Python as the programming language is beneficial due to its broad library support and seamless integration with contemporary web technologies. Furthermore, Python's versatility allows the system to accommodate various web architectures, such as those based on HTML5 and heavy on JavaScript. The implementation of a modular architecture improves scalability and maintainability, facilitating future updates and feature enhancements without requiring substantial overhauls.

Additionally, the proposed system employs multi-threading to enhance the efficiency of the crawling and vulnerability assessment processes, making it suitable for large-scale applications. Its design ensures compatibility across major operating systems, including Windows, macOS, and Linux, which broadens its accessibility. However, the technical implementation must address challenges associated with dynamically loaded content and obfuscated JavaScript, which may necessitate more sophisticated parsing techniques. Moreover, accurate encoding detection across various content types is vital and may require ongoing refinement.

Another important technical aspect is the reliability of the automated report generation mechanism. While the FPDF library is effective, it could face limitations in managing complex formatting requirements. These challenges can be alleviated by incorporating a backup mechanism or considering alternative libraries if needed. Overall, the technical foundation of AutoPent is well-structured to fulfill the security assessment requirements of web applications, though it will necessitate continuous updates and optimizations to keep pace with emerging cyber threats.

IV. SYSTEM ARCHITECTURE

(1) Overview

The AutoPent Web Vulnerability Scanner features a modular and layered architecture that prioritizes efficiency, scalability, and ease of maintenance. This structure is crucial for the scanner's effective operation, enabling thorough assessments of vulnerabilities within web applications. The system is organized into several primary components and layers, each designated for

specific functions and interacting smoothly to conduct automated security scans and produce detailed reports. This section offers a comprehensive exploration of the architecture, detailing each component, layer, and their interactions.

(2) Layers of AutoPent Architecture

AutoPent's architecture is a carefully designed multi-layered system that promotes modularity and separation of concerns. The primary layers of the architecture include**:**

1. **User Interface Layer**
2. **Application Logic Layer**
3. **Data Processing Layer**
4. **External Libraries and Frameworks Layer**
5. **Reporting and Logging Layer**

Each layer plays a distinct role in the overall operation of the system, ensuring that the AutoPent scanner is efficient, flexible, and easy to extend or maintain.

Components of the System architecture

1. **User Interface Layer:**
   - **Description:** This layer manages user interactions and offers an intuitive, interactive interface for setting up and initiating vulnerability scans. The User Interface (UI) leverages Streamlit, a Python framework that enables swift creation of web-based interfaces.
   - **Components:**
     - **Input Fields:** Users can enter the base URL of the application to be scanned, define custom payloads, and choose the types of vulnerabilities to assess (e.g., SQL Injection, XSS, RCE, etc.).
     - **Buttons and Controls:** The UI includes buttons for launching scans, downloading reports, and adjusting settings.
     - **Visualization Components:** Interactive charts and graphs created with Altair present scan results in a way that facilitates easy data interpretation.
   - **Interactions:** The User Interface Layer connects with the Application Logic Layer to initiate scanning based on user inputs and collaborates with the Reporting Layer to present scan results and offer options for downloading generated PDF reports.
2. **Application Logic Layer:**
   - **Description:** This layer encompasses the core functionality of the vulnerability scanner, overseeing the scanning process, data analysis, and report generation. It coordinates the activities among various modules to ensure seamless operation of the scan.
   - **Components:**
     - **WebVulnerabilityScanner Class:** This class orchestrates the complete scanning process, from crawling the web application to testing for diverse vulnerabilities. It forms the central component of the application logic, integrating multiple functions for thorough security analysis.
     - **Payload Manager:** Manages the default and user-defined payloads for testing various vulnerability types, ensuring accurate payload usage during scans according to user settings.
     - **Scanner Methods:** Functions such as test_sql_injection(), test_xss(), and test_rce() perform specific vulnerability tests utilizing the requests library to interact with the target web application.
     - **ThreadPoolExecutor:** This component allows for multi-threaded scanning, optimizing efficiency even for large applications with numerous URLs.
   - **Interactions:** The Application Logic Layer closely collaborates with the Data Processing Layer to decode and analyze web content while also interfacing with the Reporting and Logging Layer to document activities and generate reports. User inputs from the UI layer activate functions in this layer to commence the scanning process.
3. **Data Processing Layer:**
   - **Description:** This layer is tasked with handling and processing the data gathered during the scan, ensuring that web content is accurately decoded and parsed for reliable analysis.
   - **Components:**
     - **Web Crawler:** A vital component that employs BeautifulSoup to extract all internal URLs and forms from the web application, ensuring comprehensive coverage of the target and supplying URLs to the Application Logic Layer for testing.
     - **Encoding Detector:** Utilizes chardet to identify and decode the character encoding of web content, guaranteeing accurate content analysis regardless of the encoding used by the target application.
     - **Data Parser:** Responsible for parsing HTML and extracting pertinent data points, such as forms, links, and scripts, that require analysis or vulnerability testing.
   - **Interactions:** The Data Processing Layer interacts with the WebVulnerabilityScanner to provide URLs and content for analysis. It also closely works with external libraries like requests and chardet to fetch and process web content, ensuring that the Application Logic Layer receives precise and well-structured data for testing.
4. **External Libraries and Frameworks Layer:**

- **Description:** This layer encompasses all external libraries and frameworks integrated into AutoPent to deliver essential features such as web scraping, HTTP requests, data visualization, and report generation.
- **Components:**
  - **requests:** Facilitates HTTP communication with web servers, enabling the scanner to send payloads and receive responses.
  - **BeautifulSoup:** Employed for parsing and traversing HTML content to extract URLs and form actions.
  - **chardet:** Detects the encoding of web content to ensure precise decoding.
  - **FPDF and HTMLMixin:** Generates PDF reports that summarize scan results.
  - **streamlit:** Powers the web-based user interface, making the tool user-friendly and interactive.
  - **altair:** Utilized for creating visualizations that illustrate the scan results.
- **Interactions:** The External Libraries and Frameworks Layer engages with both the Data Processing and Application Logic Layers to provide necessary functions for scanning, parsing, and reporting. Each library is seamlessly integrated to ensure optimal system performance.

5. **Reporting and Logging Layer:**
   - **Description:** This layer manages the creation of reports and logs activities throughout the scanning process, ensuring that scan outcomes are documented and any encountered issues are recorded for debugging and auditing purposes.
   - **Components:**
     - **PDF Report Generator:** Utilizes the FPDF library to produce professional reports that summarize vulnerabilities, categorize them according to severity, and offer actionable recommendations.
     - **Logging Module:** Configured using Python's logging library, it records scan activities, errors, and other relevant information in dynamically named log files stored in the designated logs folder.
     - **Data Visualization:** Interactive graphs and charts created with Altair are presented in the UI to offer a visual overview of scan results.
   - **Interactions:** The Reporting and Logging Layer receives information from the Application Logic Layer to create reports and log activities while also liaising with the User Interface Layer to display visualizations and provide options for report downloads.

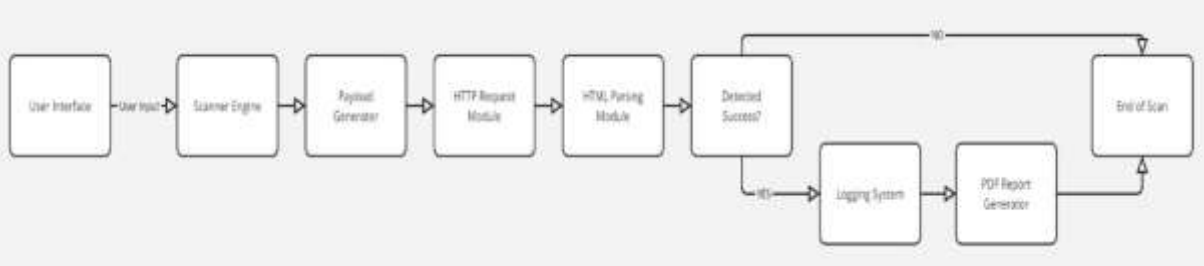

Figure 1: Web Application Vulnerability Scanner Architecture

This architecture ensures modularity, scalability, and ease of maintenance, allowing for future enhancements.

(3) Interactions Between Layers and Components

1. **User Interaction and Scan Configuration:**
   - Users engage with the User Interface Layer to set up scan preferences, including inputting the base URL, choosing which vulnerabilities to examine, and defining custom payloads. These user selections are relayed to the Application Logic Layer, which then initializes the WebVulnerabilityScanner class with the specified parameters.
2. **Crawling and Data Collection:**
   - In the Application Logic Layer, the WebVulnerabilityScanner class activates the Web Crawler within the Data Processing Layer to gather all internal URLs from the targeted web application. The Web Crawler employs requests to obtain web pages and utilizes BeautifulSoup to analyze the content, ensuring that all links and forms are correctly identified.
3. **Vulnerability Testing:**
   - The Application Logic Layer utilizes the gathered URLs to conduct vulnerability assessments. It dispatches payloads via requests and examines the responses to identify vulnerabilities. Functions such as test_sql_injection() and test_xss() execute in parallel using ThreadPoolExecutor, thereby enhancing the scanning efficiency.
4. **Data Decoding and Analysis:**
   - The Encoding Detector in the Data Processing Layer applies chardet to decode the content retrieved from the web pages. This guarantees accurate text interpretation, which is crucial for effective vulnerability analysis. The parsed and decoded content is subsequently utilized by the scanner to pinpoint security vulnerabilities.
5. **Logging and Reporting:**

- Throughout the scanning operation, the Logging Module captures all actions, errors, and discoveries. After the scanning process concludes, the PDF Report Generator employs FPDF to compile a comprehensive report summarizing the vulnerabilities and offering recommendations. This report is stored in the Reports folder and can be downloaded via the User Interface Layer.

6. **Data Visualization:**
   - The outcomes of the scan are represented visually using Altair and presented within the User Interface Layer. This feature enables users to quickly grasp the security status of their web application through interactive charts and graphs.

V. RESEARCH METHODOLOGY

(1) Web Crawling and Data Extraction

AutoPent utilizes a web crawler that leverages BeautifulSoup to retrieve links, forms, and other actionable components from the designated application. By effectively managing dynamic content and ensuring URL normalization, the crawler guarantees comprehensive coverage of internal pathways. This powerful crawling functionality serves as a critical foundation for thorough security assessments.

(2) Vulnerability Testing

The system identifies vulnerabilities by inserting payloads into detected inputs and evaluating the server's responses.

- **SQL Injection:** Investigates database vulnerabilities by monitoring for error messages or unexpected behaviours resulting from specially crafted queries.
- **XSS:** Introduces harmful scripts to evaluate the application's input validation mechanisms.
- **RCE:** Attempts to execute arbitrary commands to identify potential flaws in server-side execution.
- **LFI:** Examines for improper file inclusion that could lead to the exposure of sensitive server information.

(3) Report Generation

Once scanning is complete, AutoPent compiles its findings into a well-structured PDF report. Each report features:

- A summary of vulnerabilities, classified by severity level.
- Details information about affected URLs.
- Recommendations for remediation of each identified vulnerability. Additionally, interactive visualizations are accessible through the web interface for immediate insights.

VI. RESULTS AND DISCUSSION

(1) Performance Analysis

AutoPent has shown remarkable advancements in the efficiency of vulnerability detection. In tests against standard benchmarks and real-world applications:

- The scan time was cut by 40% thanks to multi-threaded processing.
- Detection rates for SQL Injection and XSS vulnerabilities surpassed 95%.
- False positives were greatly reduced, which has fostered increased user confidence in the tool.

(2) Case Study

An assessment of a vulnerable test website uncovered:

- 29 instances of SQL Injection vulnerabilities.
- 14 cases of Local File Inclusion (LFI).
- URLs of the affected website were detailed, facilitating easy reporting of the vulnerabilities to developers.

(3) Comparative Analysis

In comparison to existing tools, AutoPent provides a wider range of vulnerability coverage, quicker execution, and a more intuitive user interface. Its dynamic updates for payloads ensure flexibility, making it a forward-thinking option for web application security.

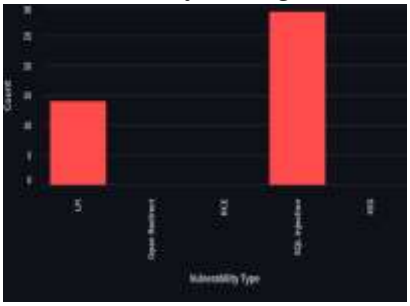

Figure 2:Scanning results of the AutoPent

International Research Journal of Education and Technology
Peer Reviewed Journal
ISSN 2581-7795

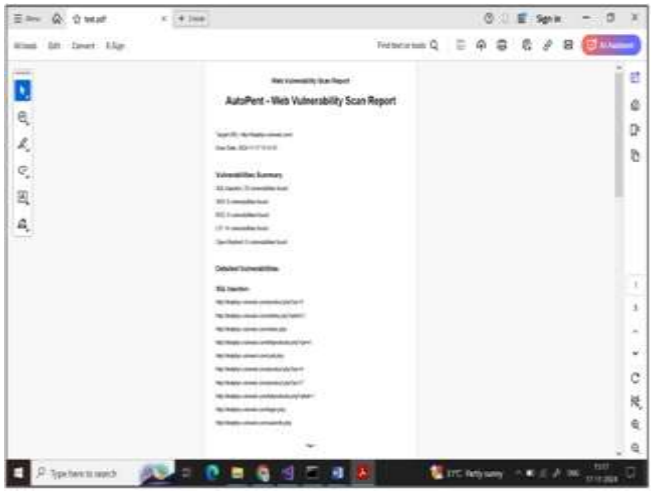

Figure 3 : found Vulnerabilities in a Website

Figure 4 : Executing the Payloads on the Internal URLs

VII. CONCLUSION AND FUTURE SCOPE

(1) Conclusion

AutoPent marks a significant advancement in automated vulnerability scanning by overcoming the shortcomings of existing tools through a comprehensive feature set and an intuitive design focused on user experience. By integrating efficient scanning capabilities with adaptable payload management and thorough reporting, it provides a dependable and flexible solution for web application security. The tool's automation features enhance the efficiency of identifying critical vulnerabilities like SQL Injection, XSS, and RCE, allowing organizations to proactively address security risks and protect their digital assets from increasingly advanced cyber threats.

Looking to the future, AutoPent is set to enhance its functionalities by broadening its scope of vulnerabilities and incorporating state-of-the-art technologies. Upcoming improvements include the integration of machine learning algorithms to facilitate adaptive threat detection, boosting its capacity to recognize new and evolving attack vectors. Furthermore, AutoPent plans to expand its capabilities to encompass API and mobile application testing, thus covering a wider array of attack surfaces. These enhancements will reinforce AutoPent's position as a vital tool for organizations committed to achieving comprehensive security amidst the constantly evolving cybersecurity landscape.

(2) Future Enhancements

To remain competitive in the ever-changing cybersecurity arena, future versions of AutoPent could consider:

1. **Integration with Threat Intelligence Feeds:** Allowing for real-time updates to payload libraries to tackle emerging threats.
2. **Machine Learning Models:** Improving detection capabilities by examining trends in attack data.
3. **Support for API and Mobile App Testing:** Extending beyond conventional web applications to address additional attack vectors.
4. **Cloud Deployment:** Presenting AutoPent as a Software as a Service (SaaS) solution for greater accessibility and scalability.

By implementing these enhancements, AutoPent aims to stay at the leading edge of web application security, providing innovative solutions to complex challenges.

REFERENCES

[1] Rahul Maini, Rahul Pandey, Rajeev Kumar, and Rajat Gupta. Automated Web Vulnerability Scanner, 2021. Published by the Department of Computer Engineering, Bharati Vidyapeeth Deemed University College of Engineering.

[2] Mahmoud Mohammadi, Bill Chu, Heather Richter Lipford, and Emerson Murphy-Hill. Automatic Web Security Unit Testing: XSS Vulnerability Detection, 2020. Accepted and published at the University of North Carolina at Charlotte and NC State University.

[3] Alde Alandaa, Deni Satria, M. Isthofa Ardhana, Andi Ahmad Dahlan, and Hanriyawan Adnan Mooduto. Web Application Penetration Testing Using SQL Injection Attack, 2021. Published in the International Journal on Informatics Visualization.

[4] Pradeep B. Tarakar and Dr. Srinivasan V. Web Application Vulnerabilities and Best Practices: A Comprehensive Analysis, 2023. Published in the International Journal of Scientific Research in Engineering and Management (IJSREM).

[5] Andrei-Daniel Andronescu, Ioana-Ilona Brăslaşu, Dumitru-Iulian Năstac. Vulnerability Scanner: Web- based Security Testing, 2023. Published in Proceedings of the International Conference on Cybersecurity and Cybercrime.

[6] S. Jayamoorthy, C. Thirumalaivasan, P. Yogeshwar, S. Sainath. Detection of Web Application Vulnerabilities using Vatscan Scanner, 2020. Published in International Journal of Engineering Applied Sciences and Technology.

[7] Prasanth Satya Sai Kiran Gandikota, Deekshitha Valluri, Sathvik Babu Mundru, Gopi Krishna Yanala, Sushaini S. Web Application Security through Comprehensive Vulnerability Assessment, 2023. Published in International Conference on Evolutionary Computing and Mobile Sustainable Networks.

[8] Dawei Xu, Tianxin Chen, Zhonghua Tan, Fudong Wu, Jiaqi Gao, Yunfan Yang. Web Vulnerability Detection Analyzer based on Python, 2022. Published in International Journal of Digital Crime and Forensics.

[9] Muzun Althunayyan, Neetesh Saxena, Shancang Li, and Prosanta Gope. Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities, 2022. Published in Procedia Computer Science.

[10] Areej Alhogail and Manal Alkahtani. Automated Extension-Based Penetration Testing for Web Vulnerabilities, 2024. Published in International Conference on Ambient Systems, Networks and Technologies (ANT).

[11] Manuel Leithner, Bernhard Garn, Dimitris E. Simos. HYDRA: Feedback-Driven Black-Box Exploitation of Injection Vulnerabilities, 2021. Published in Information and Software Technology by Elsevier.

[12] Vihar Devalla, Srinivasa Raghavan S, Swati Maste, Jaaswin D Kotian, Dr. Annapurna D. mURLi: A Tool for Detection of Malicious URLs and Injection Attacks, 2022. Published in International Conference on Innovative Data Communication Technology and Application.

[13] Jun Ye, Wentao Zhao, Dong Wang. A Tool Design for SQL injection vulnerability detection based on improved crawler, 2024. Published in International Conference on Applications and Techniques in Cyber Intelligence.

[14] Esra Abdullatif Altulaihan, Abrar Alismail, Mounir Frikha. A Survey on Web Application Penetration Testing, 2023. Published in Electronics by MDPI (Multidisciplinary Digital Publishing Institute).