**Web Application Security Testing: A NIDS-based Penetration Testing Approach**

## A PROJECT REPORT

***Submitted by,***

|  |  |  |
| --- | --- | --- |
| SHREYA MALSHETTY |  | ID No: 20211CCS0061 |
| JAYALAKSHMI D |  | ID No: 20211CCS0064 |
| CHAITHRA BANGER |  | ID No: 20211CCS0114 |
| SRUJANA MNAJUNATH |  | ID No: 20211CCS0116 |
|  |  |  |

### *Under the guidance of,*

**Dr. Sharmasth Vali Y  
Associate Professor  
Department of Computer Science Engineering in *partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING, COMPUTER ENGINEERING, INFORMATION SCIENCE AND ENGINEERING Etc.**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**DECEMBER 2024**

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**CERTIFICATE**

This is to certify that the Project report **“Web Application Security Testing: A NIDS-based Penetration Testing Approach”** being submitted by “SHREYA MALSHETTY; JAYALAKSHMI D; CHAITHRA H B; SRUJANA MANJUNATH” bearing roll number(s) “20211CCS0061; 20211CCS0064; 20211CCS0114; 20211CCS0116 ” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a Bonafide work carried out under my supervision.

|  |  |
| --- | --- |
| **Dr. Sharmasth Vali Y Associate Professor Department of Computer Science Engineering**  School of CSE&IS  Presidency University | **Dr. Anandaraj S P**  **Prof & HOD**  School of CSE&IS  Presidency University |

|  |  |  |
| --- | --- | --- |
| **Dr. L. SHAKKEERA**  Associate Dean  School of CSE  Presidency University | **Dr. MYDHILI NAIR**  Associate Dean  School of CSE  Presidency University | **Dr. SAMEERUDDIN KHAN**  Pro-Vc School of Engineering  Dean -School of CSE&IS  Presidency University |

**PRESIDENCY UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **Web Application Security Testing: A NIDS-based Penetration Testing Approach**

in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of**,** **Dr. Sharmasth Vali Y, Associate Professor,**

**School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | | SHREYA MALSHETTY |  | ID No: 20211CCS0061 | | JAYALAKSHMI D |  | ID No: 20211CCS0064 | | CHAITHRA BANGER |  | ID No: 20211CCS0114 | | SRUJANA MNAJUNATH |  | ID No: 20211CCS0116 | |

**ABSTRACT**

Imagine receiving an email from cybercriminals stating that all your personal information has been compromised—name, date of birth, home address, and finances. They are demanding money from you in exchange for not leaking your sensitive information. It's a terrifying situation to be in, isn't it? We adopt a NIDS-based approach for web application penetration testing in order to resolve this issue. Web application penetration testing are ongoing security evaluations that mimic actual attacks to evaluate how secure web applications are. The main objective is to find any possible flaws, configuration errors, or vulnerabilities that malicious users can use to jeopardize a web application's availability, confidentiality, or integrity. An NIDS is deployed to detect fraudulent activities at the network level which can complement conventional penetration testing techniques that concentrate on flaws at the software. The goal of this research is to improve the identification of security vulnerabilities at the network and application levels by combining traditional web application penetration testing with Network Intrusion Detection Systems (NIDS).

**ACKNOWLEDGEMENT**

First, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair,** School of Computer Science Engineering & Information Science, Presidency University, and “Dr. Anandaraj SP”, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Sharmasth Vali Y**, Associate Professor and Reviewer **Mr. Tanveer Ahmed-Assistant.Prof**, School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman,** department Project Coordinators “NAME” and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**SHREYA MALSHETTY**

**JAYALASKSHMI D**

**CHAITHRA H B**

**SRUJANA MNAJUNTAH**

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL NO.** | **FIGURE NAME** | **CAPTION** | **PAGE NO.** |
| 1 | Figure 6 | Network Penetration Testing | 18 |
| 2 |  | APPENDIX-B  SCREENSHOTS | 35 |
|  | Fig i & ii | NESSUS | 35 |
|  | Fig iii & iv | NMAP | 36 |
|  | Fig v | OWSAP ZAP | 37 |
|  |  |  |  |
|  |  |  |  |

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** | |
|  | **ABSTRACT** | **v** | |
|  | **ACKNOWLEDGEMENT** | **vi** | |
|  | **LIST OF FIGURES** | **vii** | |
|  | **TABLE OF CONTENTS** | **vii** | |
| **1** | **INTRODUCTION** | **1** | |
|  | 1.1 Overview of Web Application Security | 1 | |
|  | 1.1.1 Importance of Web application security in Digital age | 2 | |
|  | 1.1.2 Common types of Vulnerabilities in Web application | 3 | |
|  | 1.1.3 Risks associated with Insecure Web Application | 4 | |
|  | 1.2 Integration of NIDS with Penetration Testing | 4 | |
|  | 1.2.1 How NIDS Enhances Penetration Testing | 4 | |
|  | 1.2.2 Best practices for Integrating NIDS with Penetration Testing | 5 | |
|  | 1.3 Key Steps in NIDS Penetration Testing Process | 6 | |
| **2** | **LITERATURE SURVEY** | **9** | |
| **3** | **RESEARCH GAPS OF EXISTING METHODS** | **10** | |
|  | 3.1 Detection of Encrypted Traffic | 10 | |
|  | 3.2 False Positives and False Negatives in NIDS Alerts | 10 | |
|  | 3.3 Scalability of NIDS in Large-Scale  Web Application | 11 | |
|  | 3.4 Detection of Zero-Day Attack | 11 | |
|  | 3.5 Automated Exploitation Detection | 12 | |
|  | 3.6 User Behaviour and Analytics and Insider Threats | 12 | |
|  | 3.7 Lack of Contextual Awareness in NIDS | 12 | |
|  | 3.8 Real-Time Threat Intelligence Detection | 13 | |
| **4** | **PROPOSED METHODOLOGY** | **14** | |
|  | 4.1 Planning and Preparation | 14 |
|  | 4.2 NIDS Configuration and Monitoring Setup | 14 |
|  | 4.3 Vulnerability Scanning | 15 |
|  | 4.4 Automated Security Testing | 15 |
|  | 4.5 Manual Testing | 15 |
|  | 4.6 Exploitation | 16 |
|  | 4.7 Post-exploitation | 16 |
|  | 4.8 Reporting | 16 |
| **5** | **OBJECTIVES** | **17** |
| **6** | **SYSTEM DESIGN AND IMPLEMENTATION** | **18** |
| **7** | **TIMELINE FOR EXCECUTION OF PROJECT (GANTT CHART)** | **19** |
| **8** | **OUTCOMES** | **20** |
|  | 8.1 Enhanced Security Posture | **20** |
|  | 8.1.1 Improved Vulnerability Detection | **20** |
|  | 8.1.2 Early Detection of Threats | **20** |
|  | 8.1.3 Real-Time Alerts | **20** |
|  | 8.2 Strengthened Incident Response | **21** |
|  | 8.2.1 Improved Response Time | **21** |
|  | 8.2.2 Actionable Incident Data | **21** |
|  | 8.2.3 Automated Response | **21** |
|  | 8.3 Optimized NIDS Configuration and Performance | **21** |
|  | 8.3.1 Refined Detection Rules | **21** |
|  | 8.3.2 Reduced False Positives and Negatives | **21** |
|  | 8.4 Comprehensive Vulnerability Management | **21** |
|  | 8.4.1 Prioritized Vulnerabilities | **121** |
|  | 8.4.2 Continuous Vulnerability Monitoring | **22** |
|  | 8.5 Documentation and Reporting | **22** |
|  | 8.5.1 Detailed Penetration Testing Reports | **22** |
|  | 8.5.2 Recommendations for Future Security Improvements | **22** |
| **9** | **RESULTS AND DISCUSSIONS** | **23** |
|  | 9.1 Results: Key Outcomes Achieved | **23** |
|  | 9.1.1 Improved Detection of Vulnerabilities | **23** |
|  | 9.1.2 Real-Time Threat Detection | **24** |
|  | 9.1.3 Improved Incident Response Time | **24** |
|  | 9.14 Optimization of NIDS Detection Rules | **25** |
|  | 9.2 Discussions: Insights Gained from the Integration | **25** |
|  | 9.2.1 Synergy between Penetration Testing and NIDS | **25** |
|  | 9.2.2 Improved Vulnerability Management and Remediation | **26** |
|  | 9.2.3 Strengthening Incident Response and Threat Mitigation | **26** |
|  | 9.2.4 Challenges Encountered | **27** |
|  | 9.2.5 Continuous Improvement | **27** |
| **10** | **CONCLUSION** | **29** |
|  | **REFERENCES** | **30** |
|  | **APPENDIX-A**  **PSUEDOCODE** | **33** |
|  | **APPENDIX-B**  **SCREENSHOTS** | **35** |
|  | **APPENDIX-C**  **ENCLOSURES** | **38** |

**CHAPTER-1**

**INTRODUCTION**

Web application security testing is an essential practice for identifying vulnerabilities, ensuring the security and integrity of web applications, and preventing unauthorized access or exploitation. Given the critical role that web applications play in modern digital ecosystems, securing these applications is paramount to protecting sensitive data, ensuring privacy, and maintaining business operations. With the increasing sophistication of cyber-attacks, traditional testing methods are no longer sufficient on their own to identify and mitigate emerging threats.

One of the advanced methodologies in web application security testing is the use of Network Intrusion Detection Systems (NIDS) along with penetration testing. By leveraging NIDS, penetration testers can gain deeper insights into network traffic and identify threats that may otherwise go unnoticed, especially those targeting the underlying infrastructure and communication channels.

This approach not only helps in identifying vulnerabilities but also provides actionable intelligence for mitigating risks in real-time. By adopting a NIDS-based penetration testing strategy, organizations can better anticipate potential attacks and reinforce their web applications with stronger, more proactive security measures.

* 1. **Overview of Web Application Security**

As the world becomes more digitally connected, web applications play a crucial role in delivering services, storing sensitive data, and enabling business operations. Web application security refers to the practices, methodologies, and tools used to protect web applications from security threats, vulnerabilities, and attacks. However, this increased reliance on web-based services also makes them an attractive target for attackers. Therefore, ensuring the security of web applications is critical to prevent data breaches, financial losses, and damage to reputation.

In this section, we’ll dive into several key aspects of web application security, including its importance, common vulnerabilities, and risks associated with insecure applications.

* + 1. **Importance of Web Application Security in Digital Age**

Web applications are the backbone of most online services, from e-commerce and social media platforms to banking and healthcare applications. They allow users to interact with systems and access information over the internet. As businesses increasingly rely on web applications for operations, security becomes a fundamental concern. The main reasons for prioritizing web application security are:

* **Protection of Sensitive Data:** Web applications often store and process sensitive user data, including personal identification information (PII), financial records, and login credentials. A breach of this data can lead to identity theft, financial fraud, and regulatory consequences.
* **Ensuring Business Continuity:** Security vulnerabilities in web applications can disrupt business operations, either by enabling cyberattacks like Distributed Denial of Service (DDoS) attacks or through exploitation of flaws that allow attackers to hijack services or disable systems.
* **Protecting Brand Reputation:** A data breach or compromised web application can lead to a loss of customer trust, damaged brand reputation, and long-term financial impacts.
  + 1. **Common types of Vulnerabilities in Web application**

Web application vulnerabilities are weaknesses or flaws in the design, implementation, or configuration of an application that can be exploited by attackers. Some of the most common vulnerabilities include:

* **SQL Injection (SQLi) -** SQL Injection is one of the oldest and most dangerous types of web application vulnerabilities. It occurs when an attacker is able to inject malicious SQL code into an input field (e.g., search bars, login forms) that is then executed by the database.
* **Cross-Site Scripting (XSS) -** Cross-Site Scripting (XSS) occurs when an attacker injects malicious scripts (usually JavaScript) into a trusted website or web application. These scripts are executed in the browser of unsuspecting users.
* **Cross-Site Request Forgery (CSRF) -** CSRF exploits the trust a web application has in the user's browser. It involves tricking the user into submitting a request (such as changing their account settings) without their knowledge, leveraging their active session.
* **Broken Authentication -** Broken authentication vulnerabilities occur when an application’s authentication mechanisms (login systems, password resets, session management) are weak or improperly implemented.
* **Security Misconfiguration -** Security misconfigurations can occur in any layer of an application stack (e.g., the server, database, application, cloud services) when they are incorrectly configured or left with default settings.
* **Sensitive Data Exposure -** Sensitive data exposure happens when an application does not adequately protect sensitive data, such as passwords, credit card numbers, or health records, during storage or transmission.
  + 1. **Risks associated with Insecure Web applications**
* **Data Breaches**
* **Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks**
* **Malware Injection**
* **Reputation Damage**

**1.2 Integration of NIDS with Penetration testing**

The integration of **Network Intrusion Detection Systems (NIDS)** with **penetration testing** creates a robust, multi-layered security approach that enhances the ability to detect, analyze, and respond to potential threats in real time. While penetration testing helps uncover vulnerabilities in web applications through simulated attacks, NIDS focuses on continuously monitoring network traffic to detect malicious activities. Combining these two methodologies creates a more comprehensive security posture for web applications, providing both proactive testing and real-time detection of threats.

**1.2.1. How NIDS Enhances Penetration Testing**

* **Real-Time Detection of Exploited Vulnerabilities -** During penetration testing, ethical hackers attempt to exploit vulnerabilities. NIDS can monitor this activity and provide real-time alerts on the actions of the penetration testers.
* **Automating and Streamlining the Penetration Testing Process -** NIDS can be integrated with automated vulnerability scanners, providing **continuous testing** of the web application, which can be especially useful in DevOps environments.

**1.2.2 Best practices for Integrating NIDS with Penetration Testing**

* **Clear Objectives and Scope:** Ensure that both penetration testing and NIDS are aligned in terms of goals and scope. This ensures that penetration tests focus on relevant vulnerabilities and that NIDS are set up to detect the associated attack patterns.
* **Regularly Update and Calibrate NIDS:** Keep NIDS updated with the latest threat intelligence and patterns identified in penetration testing. This helps improve the accuracy of attack detection.
* **Automate Alerting and Reporting:** Integrate automated tools to streamline the alerting process, ensuring that penetration testers and security teams receive timely notifications about suspicious activity or successful exploits.
* **Continuous Monitoring and Testing:** Leverage continuous monitoring with NIDS while conducting regular penetration tests to ensure the security posture evolves with the changing threat landscape.
* **Comprehensive Post-Test Reporting:** Combine findings from both NIDS and penetration testing in detailed reports that cover both the discovered vulnerabilities and active exploitation attempts, with actionable remediation advice.

**1.3 Key steps in NIDS-Based Penetration Testing Process**

**Step 1: Define Testing Objectives and Scope -** Before beginning any testing activities, it is essential to define the objectives and scope of the penetration test, as well as the integration of NIDS. Some key considerations include:

* **Testing Scope:** Identify which systems, web applications, and network segments are part of the testing process. This may involve specific applications, databases, or infrastructure components.
* **Goals:** Determine the goals of the penetration test, such as identifying vulnerabilities in the application, testing for unauthorized access, or evaluating the effectiveness of existing security measures (including NIDS).
* **NIDS Focus Areas:** Identify areas where NIDS will focus its monitoring efforts, such as suspicious login attempts, access to sensitive data, or unusual outbound traffic indicative of a data exfiltration attempt.

**Step 2: Perform Network Mapping and Vulnerability Assessment -** Before executing the penetration test, the network infrastructure and web application must be mapped to identify potential vulnerabilities. This includes:

* **Network Discovery:** Use tools to map the network, identify active devices, services, and open ports. This can help identify points of potential exploitation.
* **Vulnerability Scanning:** Conduct a vulnerability scan to find weak spots, misconfigurations, and known vulnerabilities that could be exploited during the penetration test. Tools such as Nessus or OpenVAS may be used to automate this process.
* **Configuration of NIDS:** Configure the NIDS to monitor the identified network segments and vulnerable systems. NIDS should be set to watch for common attack patterns, such as SQL injections, buffer overflows, and DDoS attempts, based on the identified vulnerabilities.

**Step 3: Execute Penetration Testing -** With the NIDS configured and the scope defined, the next step is to execute the penetration test. This involves simulating real-world attacks on the web application, network, or system to discover potential vulnerabilities. Penetration testers may use a variety of techniques:

* **Reconnaissance and Information Gathering:** Gather information about the target, such as its domain names, IP addresses, and public-facing services.
* **Exploitation:** Attempt to exploit vulnerabilities identified during the vulnerability assessment. Common penetration testing techniques include SQL injection, cross-site scripting (XSS), command injection, and privilege escalation.
* **Lateral Movement:** After gaining initial access, attempt to move laterally within the network to escalate privileges or access more sensitive data.
* **Post-Exploitation:** Simulate the actions of an attacker who has compromised the system, including attempting to extract sensitive data or maintain persistence.

During this phase, the NIDS actively monitors the network traffic generated by the penetration testers' activities. The NIDS is designed to detect any malicious activity and provide real-time alerts, helping security teams understand how well the system can detect and respond to attacks.

**Step 4: Monitor and Analyse NIDS Alerts -** While the penetration test is ongoing, the NIDS continuously monitors the network for suspicious activity. Key tasks during this phase include:

* **NIDS Alert Generation:** As penetration testers attempt to exploit vulnerabilities, the NIDS should detect patterns indicative of attacks and generate alerts. This could include signs of SQL injection attempts, unusual login patterns, or attempts to exploit known vulnerabilities.
* **Alert Analysis:** The security team should analyse the NIDS alerts in real-time to evaluate their accuracy. This analysis will determine whether the NIDS correctly detected the penetration testers’ simulated attack and if there are any false positives or missed detections.
* **Incident Response Simulation:** Based on the NIDS alerts, security teams should practice incident response. This may involve isolating affected systems, blocking malicious IP addresses, or taking other corrective actions.

**Step 5: Post-Test Analysis and Reporting** - Once the penetration testing and NIDS monitoring are complete, the next step is to analyse the results and generate a comprehensive report. This includes:

* **Penetration Test Findings:** Document the vulnerabilities and weaknesses discovered during the test, including any exploited vulnerabilities, attack vectors, and compromised systems.
* **NIDS Performance Evaluation:** Evaluate the effectiveness of the NIDS in detecting the simulated attacks. This includes analyzing the quality of the alerts (e.g., true positives vs. false positives) and the system's ability to detect sophisticated attack techniques.
* **Remediation Recommendations:** Provide recommendations to fix vulnerabilities identified in the penetration test and improve NIDS configuration to enhance threat detection.
* **Security Improvements:** Suggest improvements to the overall security posture, such as patching vulnerabilities, tightening access controls, or enhancing network monitoring.

**Step 6: Retesting and Continuous Monitoring -** After addressing the findings from the penetration test and optimizing the NIDS configuration, it’s important to conduct a **retest** to verify that vulnerabilities have been resolved and that the NIDS can accurately detect any new threats. This process includes:

* **Retesting Vulnerabilities:** Test any vulnerabilities that were remediated during the previous phase to ensure they are no longer exploitable.
* **NIDS Tuning:** Refine NIDS detection rules based on the penetration test findings to reduce false positives and improve detection accuracy.
* **Ongoing Monitoring:** Continue using NIDS to monitor network traffic and detect any suspicious activities on an ongoing basis, ensuring the network remains secure.

**CHAPTER-2**

**LITERATURE SURVEY**

Web application security testing is essential for identifying vulnerabilities in digital platforms, with penetration testing playing a critical role in this process. A NIDS (Network Intrusion Detection System)-based approach enhances security testing by monitoring network traffic for suspicious activity, integrating seamlessly into various testing frameworks.

*"Impact and Research Challenges of Penetrating Testing and Vulnerability Assessment on Network Threat”* - The author makes the point that security is a significant problem and to overcome that they have proposed vulnerability assessment and penetration testing for network-based vulnerability. Vulnerability threats (such as viruses, trojans, and worms), network problems (such as false positives, negatives, and system complexity), and pen testing risks (such as disruption, ethical and legal issues). Additionally, they have contained future work that can be done in an IoT, cloud, or aiml-based manner.

*"Network Intrusion Detection System using Deep Learning”* -This paper proposes the use of deep learning architectures to develop an adaptive and resilient network intrusion detection system (IDS) to detect and classify network attacks they used the UNSW-NB15 dataset. Their suggested method achieved 95.4% and 95.6% total accuracy. Future work calls for transfer learning along with boot strapping techniques.

**CHAPTER 3**

**RESEARCH GAPS OF EXISTING METHODS**

While existing methods for web application security testing, including traditional penetration testing and the integration of Network Intrusion Detection Systems (NIDS), offer significant insights into potential vulnerabilities, there are several research gaps that need further exploration and innovation. These gaps typically relate to the limitations of current techniques, the evolving nature of threats, and the technological advancements that could enhance security testing. Identifying these gaps can help guide future research and improve the effectiveness of web application security practices.

**3.1 Detection of Encrypted Traffic:**

*- Current Limitation:* Modern web applications increasingly rely on encrypted communication (e.g., HTTPS) to protect data in transit. NIDS often struggles to inspect encrypted traffic, limiting its ability to detect threats like malware, data exfiltration, or man-in-the-middle attacks.

*- Research Gap:* Developing methods for inspecting encrypted traffic without compromising privacy and security is a major challenge. Techniques like advanced SSL/TLS inspection or leveraging machine learning to identify patterns in encrypted traffic could be further explored.

**3.2 False Positives and False Negatives in NIDS Alerts:**

*- Current Limitation:* NIDS are prone to generating false positives (alerting on harmless activities) and false negatives (failing to detect actual attacks). This compromises the accuracy and reliability of the security testing process.

*- Research Gap:* There is a need for improved algorithms and machine learning techniques that can reduce the occurrence of false positives and false negatives. Research into more adaptive NIDS systems that can learn from new attack patterns and network behavior is crucial.

**3.3 Scalability of NIDS in Large-Scale Web Applications:**

**-**  *Current Limitation:* NIDS can experience performance degradation when deployed in large-scale networks with high traffic volume. This limits their ability to provide real-time monitoring for complex, high-traffic web applications.

*- Research Gap:* Developing scalable NIDS solutions capable of analyzing vast amounts of network traffic in real-time without sacrificing performance is an ongoing challenge. Distributed NIDS architectures or more efficient traffic analysis algorithms could improve scalability.

**3.4 Detection of Zero-Day Attacks:**

*- Current Limitation:* Existing penetration testing and NIDS-based approaches often struggle to detect zero-day vulnerabilities—previously unknown exploits that attackers can use to compromise an application.

- *Research Gap:* More research is needed in the development of proactive techniques such as anomaly-based detection, behavior analysis, and machine learning to detect previously unknown or unpatched vulnerabilities in real-time.

**3.5 Automated Exploitation Detection:**

***-*** *Current Limitation:* While penetration testing often identifies vulnerabilities, the automated detection of exploits using NIDS is still an underdeveloped area. Many existing methods do not efficiently correlate vulnerabilities with actual exploitation attempts.

*- Research Gap:* The ability to automate the detection of exploit attempts using NIDS (e.g., correlating vulnerability scans with live attack traffic) could significantly improve the accuracy and speed of response during penetration tests.

**3.6 User Behavior Analytics and Insider Threats:**

***-*** *Current Limitation:* NIDS primarily focuses on network traffic and external threats, often overlooking the risk posed by insider threats or anomalous user behavior within a web application.

***-*** *Research Gap:* Researching and integrating User and Entity Behavior Analytics (UEBA) within NIDS could provide deeper insights into potential insider threats and unusual user activities that traditional intrusion detection might miss.

**3.7 Lack of Contextual Awareness in NIDS:**

**-** *Current Limitation:* Many NIDS tools lack contextual awareness, meaning they may miss attacks that appear legitimate in isolation but are suspicious when considered in the broader context of application behavior or business processes.

***-*** *Research Gap:* Developing NIDS that can understand the context of web application traffic, such as the user roles, session data, or business workflows, could help identify more sophisticated attacks (e.g., business logic flaws, privilege escalation).

**3.8 Real-time Threat Intelligence Integration:**

**-** *Current Limitation:* While NIDS can detect attacks based on predefined signatures or patterns, they often lack the ability to dynamically integrate up-to-date threat intelligence to detect emerging attack methods.

*Research Gap:* Research into integrating real-time threat intelligence feeds into NIDS, enabling more adaptive and context-aware security testing, could significantly improve detection capabilities and speed of response.

**CHAPTER-4**

**PROPOSED MOTHODOLOGY**

Here’s a proposed methodology for conducting web application penetration testing using a network intrusion detection system (NIDS)-based approach. This methodology leverages **NMAP, OWASP ZAP, Nessus, and Metasploit** to ensure a thorough evaluation of the web application's security posture.

*4.1 Planning and Preparation:*

*-* Define Scope: Identify the web application’s boundaries, including subdomains, APIs, and third-party services.

*-* Gathering information - NMAP: Use NMAP to perform reconnaissance on the target web application. This includes scanning for open ports, services, and potential vulnerabilities.

Command example: **map -sS -p- -A <target\_ip\_or\_domain>**

*-* Understand the Architecture: Review application architecture and identify components such as servers, databases, and other integrated services.

*4.2 NIDS Configuration and Monitoring Setup:*

*-* Set Up NIDS: Configure a NIDS (such as Nmap or Suricata) to monitor network traffic during the penetration test.

*-* Traffic Logging: Ensure that all network traffic during the testing period is logged for later analysis.

*4.3 Vulnerability Scanning:*

*-* Use Nessus: Conduct a comprehensive vulnerability scan using Nessus to identify known vulnerabilities in the web application and its components.

- Focus on critical vulnerabilities that could be exploited.

- Generate a report with findings and prioritize based on risk.

*4.4 Automated Security Testing:*

*-* OWASP ZAP:Launch OWASP ZAP to perform automated security testing on the web application.

- Use the Active Scan feature to find common vulnerabilities such as SQL injection, cross-site scripting (XSS), and more.

- Set ZAP to run in conjunction with the NIDS to capture all the network traffic generated during the scan.

- Generate a report of the vulnerabilities detected by ZAP for further analysis.

*4.5 Manual Testing:*

*-* Conduct Manual Tests:After automated testing, perform manual testing to identify vulnerabilities that automated tools may have missed. Focus on:

- Business logic flaws

- Session management issues

- Insecure direct object references

- Use ZAP’s Manual Explore mode for additional insights.

*4.6 Exploitation:*

*-* Metasploit Framework:

- Use Metasploit to attempt to exploit identified vulnerabilities.

- Focus on vulnerabilities confirmed during the scanning phase and manual testing.

- Execute penetration tests such as SQL injection or command injection to assess the impact.

- Ensure to log all activities and payloads executed for later review.

- Command example**: MSF console** and use various payloads against identified vulnerabilities.

4.7 Post-exploitation:

- Assess Impact: Determine the potential impact of successfully exploiting vulnerabilities, such as data leakage, privilege escalation, or complete system compromise.

- Data Collection: Collect any sensitive data obtained during exploitation and evaluate the security controls in place.

*4.8 Reporting:*

*-* Compile Findings: Document all findings from automated and manual tests, including vulnerabilities, exploitation attempts, and evidence.

- Use tools like Nessus and ZAP to generate detailed reports.

*-* Provide Recommendations:

- Offer actionable recommendations to mitigate identified vulnerabilities.

- Include guidance on improving security posture, such as updating software, implementing WAF (Web Application Firewall) rules, and enhancing coding practices.

**CHAPTER-5**

**OBJECTIVES**

*1. Identify Vulnerabilities:*

Discover and catalog weaknesses in systems, applications, or networks that could be exploited by attackers.

*2. Test Detection Capabilities:*

Assess the effectiveness of security tools (e.g., NIDS, firewalls) in recognizing and alerting potential threats or malicious activities.

*3. Analyze Network Traffic:*

Monitor and examine data packets transmitted over a network to identify unusual patterns or signs of security breaches.

*4. Evaluate Incident Response:*

Review and assess the efficiency and effectiveness of the processes and actions taken during and after a security incident.

*5. Continuous Improvement:*

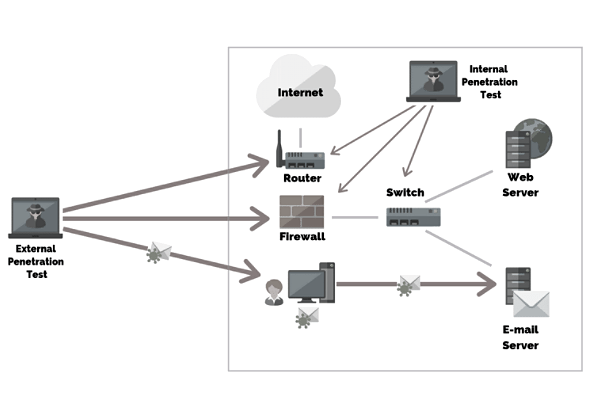
Implement ongoing enhancements to security practices, policies, and tools based on lessons learned from incidents and assessments.

*6. Documentation and Reporting:*

Record and communicate findings, incidents, and actions taken, providing a clear account of security activities and outcomes for stakeholders.

**CHAPTER-6**

**SYSTEM DESIGN & IMPLEMENTATION**

****

**Figure 6: Network Penetration Testing**

**CHAPTER-7**

**TIMELINE FOR EXECUTION OF PROJECT**

**(GANTT CHART)**

**PHASE 1:** Project Planning & Preparation

**PHASE 2:** Research & Literature Review

**PHASE 3:** Tool Selection & Configuration

**PASE 4:** Penetration Testing & NIDS Integration

**PHASE 5:** Analysis of Results

**PHASE 6:** Reporting & Recommendations

**PHASE 7:** Project Conclusion & Review

**CHAPTER-8**

**OUTCOMES**

The primary goal of this project is to improve web application security by integrating Network Intrusion Detection Systems (NIDS) with penetration testing methodologies. By leveraging the strengths of both approaches, this project aims to provide a comprehensive solution for identifying vulnerabilities, detecting real-time attacks, and enhancing overall security measures. Below are the key outcomes that can be expected from the successful execution of this project:

**8.1 Enhanced Security Posture:**

***8.1.1 Improved Vulnerability Detection***- Implement advanced scanning tools and methodologies to identify potential weaknesses in the network, applications, and systems. This includes regular automated scans and manual assessments to ensure comprehensive coverage.

***8.1.2 Early Detection of Threats*** - Utilize machine learning and behavioral analytics to detect anomalies in network traffic that may indicate a potential security breach. Early threat detection mechanisms can significantly reduce the impact of attacks.

***8.1.3 Real-Time Alerts*** - Establish a centralized alert system that provides immediate notifications to security teams regarding suspicious activities or detected threats, ensuring rapid awareness and response.

**8.2 Strengthened Incident Response:**

***8.2.1 Improved Response Times*** -Develop and implement incident response plans that streamline communication and actions among team members, reducing the time taken to react to incidents.

***8.2.2 Actionable Incident Data*** - Ensure that incident reports provide comprehensive insights, including the nature of the threat, affected systems, and potential impact, to facilitate informed decision-making during an incident.

***8.2.3 Automated Response*** *-* Implement automation tools that can take predefined actions in response to specific types of incidents, such as isolating compromised systems or blocking malicious traffic, to enhance response efficiency.

**8.3 Optimized NIDS Configuration and Performance:**

***8.3.1 Refined Detection Rules*** - Regularly review and update detection rules to ensure they are aligned with the latest threat intelligence and are effective in identifying relevant threats without overwhelming the system.

***8.3.2 Reduced False Positives and Negatives*** - Use advanced tuning techniques and feedback from security analysts to minimize false positives (legitimate traffic flagged as threats) and false negatives (actual threats not detected), improving the overall effectiveness of the NIDS.

**8.4 Comprehensive Vulnerability Management:**

***8.4.1 Prioritized Vulnerabilities*** - Establish a risk-based approach to vulnerability management that categorizes vulnerabilities according to their potential impact and exploitability, allowing for targeted remediation efforts.

***8.4.2 Continuous Vulnerability Monitoring*** - Implement ongoing monitoring processes to identify new vulnerabilities as they emerge and ensure timely updates to systems and applications to mitigate risks.

**8.5 Documentation and Reporting:**

***8.5.1 Detailed Penetration Testing Reports*** - Conduct regular penetration tests and document findings in detail, including vulnerabilities discovered, exploited paths, and the effectiveness of existing security measures.

***8.5.2 Recommendations for Future Security Improvements*** - Provide actionable recommendations based on the findings from assessments and incident analyses, outlining specific measures to strengthen security posture and enhance incident response capabilities.

**CHAPTER-9**

**RESULTS AND DISCUSSIONS**

The NIDS-Based Penetration Testing Approach integrates the strengths of Network Intrusion Detection Systems (NIDS) with penetration testing to provide a more comprehensive security framework. The results of this approach can be analyzed in terms of its effectiveness, improvements in security posture, and challenges encountered during its implementation. Below is a detailed discussion of the results achieved, the insights gained, and the lessons learned during the implementation of this approach.

**9.1. Results: Key Outcomes Achieved:**

9.*1.1 Improved Detection of Vulnerabilities:*

- Penetration Testing Results: During the penetration testing phase, several vulnerabilities were identified in the test systems. These included SQL injection, cross-site scripting (XSS), misconfigurations, and insecure access controls. These vulnerabilities were discovered through a combination of manual testing and automated vulnerability scanning tools.

- Integration with NIDS: The vulnerabilities identified in the penetration testing phase were then cross-referenced with the NIDS detection capabilities. For instance, the NIDS successfully detected SQL injection attempts in real-time by identifying unusual query patterns in the traffic. Similarly, XSS attacks were detected through abnormal data flow patterns in web application requests.

*9.1.2 Real-Time Threat Detection:*

- Alert Generation: One of the most important outcomes of this approach was the NIDS’s ability to generate real-time alerts during the penetration testing process. For example, during a simulated brute force login attack, the NIDS raised alerts as soon as it detected multiple failed logins attempts within a short time frame, indicating potential unauthorized access attempts.

- Response to Exploited Vulnerabilities: The NIDS was also able to detect attempts to exploit vulnerabilities that were previously identified during the penetration test. For example, once a remote code execution attempt was made (following a discovered vulnerability), the NIDS flagged the suspicious payload traffic, triggering an alert.

*9.1.3 Improved Incident Response Time:*

- Faster Mitigation: With the combined approach of penetration testing and NIDS monitoring, the security team was able to respond more quickly to threats. Alerts generated by the NIDS allowed the team to immediately isolate the affected systems, block malicious IP addresses, and investigate the root cause of the alerts.

- Proactive Action: In cases where the penetration testers simulated real-world attacks, the NIDS helped improve incident response times by alerting the security team before a significant exploit or breach occurred. For instance, when a denial-of-service (DoS) attack was simulated, the NIDS successfully detected the unusual traffic patterns and alerted the team, enabling them to take action and mitigate the attack before service disruption occurred.

*9.1.4 Optimization of NIDS Detection Rules:*

- Fine-Tuning Detection: Based on the penetration testing results, the NIDS detection rules were adjusted and fine-tuned. For example, custom signatures for SQL injection attacks were added to detect specific attack vectors that the penetration testing team had identified. Similarly, new rules were implemented to monitor traffic for signs of privilege escalation attempts and unauthorized access.

- Reduction in False Positives: Fine-tuning the NIDS rules led to a reduction in false positives. The system was better able to differentiate between legitimate traffic and potential threats, making it more efficient in detecting real attacks while minimizing unnecessary alerts.

**9.2. Discussions: Insights Gained from the Integration:**

*9.2.1 Synergy Between Penetration Testing and NIDS:*

- Complementary Strengths: One of the key insights from this approach was how well penetration testing and NIDS complemented each other. Penetration testing provided valuable insights into system vulnerabilities, which were then translated into specific attack signatures for the NIDS. This integration resulted in a system that could not only identify vulnerabilities proactively but also provide real-time detection of active attacks.

- Comprehensive Security Coverage: By integrating both methods, the system benefited from a holistic approach to security, where weaknesses were identified before they could be exploited (via penetration testing), and any attempts to exploit those weaknesses were promptly detected and flagged (via NIDS). This dual-layered approach to security greatly enhanced the organization’s overall defense mechanism.

*9.2.2 Improved Vulnerability Management and Remediation:*

- Prioritization of Vulnerabilities: Penetration testing helped prioritize vulnerabilities based on their exploitability, potential impact, and likelihood of being targeted. NIDS then allowed for continuous monitoring of these vulnerabilities, detecting attempts to exploit them in real-time.

- Ongoing Vulnerability Scanning: The system allowed for continuous scanning of vulnerabilities and enabled real-time patching or remediation efforts, ensuring that the network remained secure over time. The ability to monitor the exploitation of identified vulnerabilities continuously provided a dynamic layer of protection.

*9.2.3 Strengthening Incident Response and Threat Mitigation:*

- Effective Incident Response: The collaboration between penetration testers and the security team, empowered by NIDS alerts, led to faster detection and mitigation of security incidents. Early detection of lateral movement, privilege escalation, and other attack techniques provided the team with the necessary data to contain and respond to threats quickly.

- Feedback Loop: The incident response process was improved by the feedback loop between penetration testing and NIDS. Penetration testers could simulate attack scenarios that were directly aligned with real-world attack tactics, and NIDS would provide live alerts for those scenarios. This feedback loop strengthened the security posture by allowing the team to adjust security controls based on realistic, live threat data.

*9.2.4 Challenges Encountered:*

- Resource and Time Intensive: One of the key challenges faced during this process was the resource intensity of continuous monitoring by NIDS during the penetration testing phase. This approach required significant infrastructure, including dedicated personnel and systems, to manage the ongoing testing and monitoring.

- Complex Configuration: The integration of penetration testing and NIDS required extensive coordination between teams. Fine-tuning the NIDS detection rules to align with the vulnerabilities identified in the penetration testing phase took time and effort. Configuring NIDS to detect attack vectors that are specific to each application or system was complex and needed expertise.

- False Positives in NIDS Alerts: Despite fine-tuning, the NIDS still generated a few false positives, especially when new vulnerabilities or attack techniques were involved. This required manual review and adjustments to improve accuracy. It was clear that continuous tuning and maintenance of the NIDS were necessary to ensure optimal performance.

*9.2.5 Continuous Improvement:*

- Adaptive Security Posture: The project highlighted the importance of continuous improvement in the security system. As penetration testing and NIDS detection evolved, so did the organization’s defense strategy. The NIDS detection rules were updated regularly to adapt to newly discovered vulnerabilities, and the penetration testing methodology was refined to include a wider range of attack scenarios.

- Emerging Threats: The integration of penetration testing and NIDS helped identify new attack vectors and trends. For instance, during the testing phase, an advanced persistent threat (APT) was simulated, which required the team to incorporate more sophisticated detection methods into the NIDS. This ongoing adaptability ensured the organization stayed ahead of emerging threats.

**CHAPTER-10**

**CONCLUSION**

In conclusion, this project has explored the intersection of traditional penetration testing techniques and the integration of Network Intrusion Detection Systems (NIDS) to enhance the security of web applications. As the digital landscape continues to evolve, so do the methods employed by malicious actors to exploit vulnerabilities in web applications. These attacks can have devastating consequences, ranging from data breaches and financial losses to reputational damage and compliance violations.

Through the course of this project, we have emphasized the critical need for robust security testing practices, particularly focusing on how integrating NIDS with penetration testing offers a comprehensive and adaptive approach to identifying and mitigating potential security risks in real time. By combining offensive strategies (penetration testing) with defensive mechanisms (NIDS), organizations can gain a deeper understanding of vulnerabilities that may not be easily detectable through traditional testing methods alone.

Ensuring the security of web applications requires a multi-layered approach that integrates both offensive and defensive strategies. NIDS-based penetration testing is a powerful tool that enhances the ability to identify and address vulnerabilities across both application code and network traffic. By leveraging the strengths of both methodologies, organizations can strengthen their overall cybersecurity posture, proactively defend against evolving threats, and safeguard sensitive data and business operations.

**REFERENCES**

[1] Alhamed M, Rahman MMH. A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions. Applied Sciences. 2023; 13(12):6986. <https://doi.org/10.3390/app13126986>

[2] Altulaihan EA, Alismail A, Frikha M. A Survey on Web Application Penetration Testing. Electronics. 2023; 12(5):1229.

<https://doi.org/10.3390/electronics12051229>

[3] Vladimir Ciric, Marija Milosevic, Danijel Sokolovic, Ivan Milentijevic,

Modular deep learning-based network intrusion detection architecture for real-world cyber-attack simulation, Simulation Modelling Practice and Theory, Volume 133,2024,102916,ISSN 1569-190X,

<https://doi.org/10.1016/j.simpat.2024.102916>

[4] Saputra FA, Salman M, Hasim JAN, Nadhori IU, Ramli K. The Next-Generation NIDS Platform: Cloud-Based Snort NIDS Using Containers and Big Data. Big Data and Cognitive Computing. 2022; 6(1):19. <https://doi.org/10.3390/bdcc6010019>

[5] S. Kumar, S. Gupta and S. Arora, "Research Trends in Network-Based Intrusion Detection Systems: A Review," in IEEE Access, vol. 9, pp. 157761-157779, 2021. <https://doi:10.1109/ACCESS.2021.3129775>

[6] Heather Lawrence, Uchenna Ezeobi, Orly Tauil, Jacob Nosal, Owen Redwood, Yanyan Zhuang, Gedare Bloom,

CUPID: A labeled dataset with Pentesting for evaluation of network intrusion detection, Journal of Systems Architecture, Volume 129,2022,102621,ISSN13837621; <https://doi.org/10.1016/j.sysarc.2022.102621>

[7] Kumar, Satish & Gupta, Sunanda & Arora, Sakshi. (2021). Research Trends in Network-Based Intrusion Detection Systems: A Review. IEEE Access. PP. 1-1; <https://doi.org/10.1109/ACCESS.2021.3129775>

[8] Lirim Ashiku, Cihan Dagli, Network Intrusion Detection System using Deep Learning, Procedia Computer Science, Volume 185, 2021, Pages 239247, ISSN 1877-0509; <https://doi.org/10.1016/j.procs.2021.05.025>

[9] Vanin P, Newe T, Dhirani LL, O’Connell E, O’Shea D, Lee B, Rao M. A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning. Applied Sciences. 2022; 12(22):11752. <https://doi.org/10.3390/app122211752>

[10] Vegesna, Vinod Varma, Utilising VAPT Technologies (Vulnerability Assessment & Penetration Testing) as a Method for Actively Preventing Cyberattacks (October 25, 2023). International Journal of Management, Technology And Engineering | Volume XII, Issue VII, JULY 2022, Available at SSRN: [ssrn.com/abstract=4612524](https://doi.org/ssrn.com/abstract=4612524)

[11] Ke Chen, Man Zhang, Rufeng Liang, Junhan Chen, Jin Peng, Xun Huang, Research on the Application of Penetration Testing Frameworks in Blockchain Security, Computational and Experimental Simulations in Engineering, <https://doi.org/10.1007/978>

[12] Fredrik Heiding, Emre Süren, Johannes Olegård, Robert Lagerström,

Penetration testing of connected households,Computers & Security,Volume 126, 2023,103067,ISSN 0167-4048; <https://doi.org/10.1016/j.cose.2022.103067>

[13] Altulaihan, E.A.; Alismail, A.; Frikha, M. A Survey on Web Application Penetration Testing. Electronics 2023, 12, 1229. <https://doi.org/10.3390/electronics12051229>

[14] A. Goutam and V. Tiwari, "Vulnerability Assessment and Penetration Testing to Enhance the Security of Web Application," 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2019, pp. 601-605, doi: 10.1109/ISCON47742.2019.9036175.

[15] P. Vats, M. Mandot and A. Gosain, "A Comprehensive Literature Review of Penetration Testing & Its Applications," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 674-680, doi: 10.1109/ICRITO48877.2020.9197961.

[16] Abdulghaffar, K.; Elmrabit, N.; Yousefi, M. Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners. Computers 2023, 12, 235. <https://doi.org/10.3390/computers12110235>

[17] Soroush M. Sohi, Jean-Pierre Seifert, Fatemeh Ganji, RNNIDS: Enhancing network intrusion detection systems through deep learning,Computers & Security,Volume 102,2021,102151,ISSN 0167-4048, <https://doi.org/10.1016/j.cose.2020.102151>

[18] Ms Khushnaseeb Roshan, Aasim Zafar, Boosting robustness of network intrusion detection systems: A novel two phase defense strategy against untargeted white-box optimization adversarial attack, Expert Systems with Applications,Volume 249, Part A,2024,123567,ISSN 09574174, <https://doi.org/10.1016/j.eswa.2024.123567>.

[19] Murat Aydos, Çiğdem Aldan, Evren Coşkun, Alperen Soydan,Security testing of web applications: A systematic mapping of the literature,Journal of King Saud University - Computer and Information Sciences,Volume 34, Issue 9,2022,Pages 6775-6792,ISSN 13191578, <https://doi.org/10.1016/j.jksuci.2021.09.018>.

[20] Tadhani, J.R., Vekariya, V., Sorathiya, V. et al. Securing web applications against XSS and SQLi attacks using a novel deep learning approach. Sci Rep 14, 1803 (2024). <https://doi.org/10.1038/s41598-023-48845-4>

[21] Aljebry, A.F., Alqahtani, Y.M., Sulaiman, N. (2022). Analyzing Security Testing Tools for Web Applications. In: Khanna, A., Gupta, D., Bhattacharyya, S., Hassanien, A.E., Anand, S., Jaiswal, A. (eds) International Conference on Innovative Computing and Communications. Advances in Intelligent Systems and Computing, vol 1387. Springer, Singapore. <https://doi.org/10.1007/978-981-16-2594-7_34>

[22] Rajić, B., Stanisavljević, Ž. & Vuletić, P. Early web application attack detection using network traffic analysis. Int. J. Inf. Secur. 22, 77–91 (2023). <https://doi.org/10.1007/s10207-022-00627-1>

[23] Ayyagari, M.R., Kesswani, N., Kumar, M. et al. Intrusion detection techniques in network environment: a systematic review. Wireless Netw 27, 1269–1285 (2021). <https://doi.org/10.1007/s11276-020-02529-3>

[24] Ali, M., Haque, Mu., Durad, M.H. et al. Effective network intrusion detection using stacking-based ensemble approach. Int. J. Inf. Secur. 22, 1781–1798 (2023). <https://doi.org/10.1007/s10207-023-00718-7>

[25] Chowdhary, A.; Jha, K.; Zhao, M. Generative Adversarial Network (GAN)-Based Autonomous Penetration Testing for Web Applications. Sensors 2023, 23, 8014. <https://doi.org/10.3390/s23188014>

**APPENDIX-A**

**PSUEDOCODE**

Here is some pseudocode to work with NIDS (Network-based Intrusion Detection System) and tools like Metasploit, Nmap, Nessus, and OWASP ZAP:

// Initialize NIDS

Initialize NIDS engine

Load NIDS rules and signatures

// Integrate with Metasploit

Connect to Metasploit framework

Authenticate with Metasploit credentials

Retrieve list of available exploits and modules

// Integrate with Nmap

Run Nmap scan to enumerate hosts and services

Parse Nmap output to identify potential vulnerabilities

// Integrate with Nessus

Connect to Nessus scanner

Authenticate with Nessus credentials

Run Nessus scan to identify vulnerabilities

Parse Nessus output to retrieve vulnerability data

// Integrate with OWASP ZAP

Connect to OWASP ZAP API

Authenticate with OWASP ZAP credentials

Run OWASP ZAP scan to identify web application vulnerabilities

Parse OWASP ZAP output to retrieve vulnerability data

// Correlate data from all tools

Correlate vulnerabilities from Nmap, Nessus, and OWASP ZAP

Prioritize vulnerabilities based on severity and risk

Create report detailing vulnerabilities and recommended remediation

// Exploit vulnerabilities with Metasploit (optional)

Select exploit from Metasploit module list

Configure exploit with target IP and vulnerability details

Run exploit to gain access to vulnerable system

// Log and alert on suspicious activity

Log all suspicious activity detected by NIDS

Alert security team of potential security incidents

Note: Exploiting vulnerabilities with Metasploit should only be done with permission from the system owner and in a controlled environment.

**APPENDIX-B**

**SCREENSHOTS**

**NESSUS**

A screenshot of a computer

Description automatically generated

**Fig i**

A screenshot of a computer

Description automatically generated

**Fig ii**

**NMAP**

A screenshot of a computer

Description automatically generated

**Fig iii**

A screenshot of a computer

Description automatically generated

**Fig iv**

**OWSAP ZAP**

A screenshot of a computer

Description automatically generated

**Fig v**

**APPENDIX-C**

**ENCLOSURES**

**1. Journal publication/Conference Paper Presented Certificates of all students.**

**2. Include certificate(s) of any Achievement/Award won in any project-related event.**

**3. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.**

**4.** **Details of mapping the project with the Sustainable Development Goals (SDGs).**