

|  |  |
| --- | --- |
| Date | 14 March, 2025 |
| Team ID | PNT2025TMID04086 |
| Project Name | **Advanced Techniques in Rule Creation for Threat Detection** |

**List of teammates:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Name | College | Contact |
| 1 | Yogi Atram | St. Vincent Pallotti College of Engineering & Technology | 8446436126 |
| 2 | Ankit Singh | St. Vincent Pallotti College of Engineering & Technology | 9307192935 |
| 3 | Sambhav Dhakle | St. Vincent Pallotti College of Engineering & Technology | 8530617638 |
| 4 | Aadesh Punnajwar | St. Vincent Pallotti College of Engineering & Technology | 8010529460 |

1. **INTRODUCTION**

**1.1 Project Name: Advanced Techniques in Rule Creation for Threat Detection**

* 1. **Purpose**
* **Abstract:**

This project aims to establish a **vulnerability management framework** by integrating **Nessus vulnerability scanner** with **Security Information and Event Management (SIEM) tools.** The goal is to**:  
✅ Automate vulnerability detection  
✅ Enhance security monitoring  
✅ Integrate with SIEM for real-time threat detection  
✅ Provide actionable remediation steps**

* **Scope of the Project:**

The project focuses on:

* Automating vulnerability scanning with Nessus.
* Prioritizing vulnerabilities based on risk assessment.
* Integrating results with SIEM tools like Splunk, QRadar, or ELK Stack.
* Testing remediation effectiveness through repeated assessments.

1. **IDEATION PHASE**

**2.1 Thought Behind the Project**

* **Team Brainstorming:**

**Yogi Atram**

Automate vulnerability scanning to **reduce manual security audits.**

Use **machine learning** to improve threat detection accuracy in SIEM.

**Ankit Singh**

Integrate Nessus findings with **real-time threat intelligence feeds.**

Develop **custom SIEM correlation rules** for detecting VPN/TOR-based attacks.

**Aadesh Punnajwar**

Create a **dashboard** for centralized vulnerability reporting and SIEM alerts.

Test the system against **real-world attack scenarios** to validate its effectiveness.

**Sambhav Dhakle**

Prioritize vulnerabilities **based on business impact** rather than just severity.

Implement **automated remediation tracking** for security teams.

Each idea contributes to **automating and optimizing vulnerability management**, making SIEM **more efficient and accurate** in detecting **critical security threats**.

**2.2 Features**

* **Data Collection:**
  + The system will use Nessus to perform vulnerability assessments across different systems, including servers, workstations, and network devices.
  + Data will be collected on the type, severity, and exploitability of each vulnerability, and categorized into critical, high, medium, and low risks.
  + Additional contextual data about the infrastructure, like software versions and configurations, will be gathered for a comprehensive risk assessment.

**2.3 Priority Chart**

|  |  |  |
| --- | --- | --- |
| Priority Level | Task/Feature | Reason |
| |  | | --- | |  |   High Priority | SIEM Integration with Nessus | Ensures real-time correlation of vulnerabilities with security events. |
| High Priority | Reducing False Positives | Improves SOC efficiency by minimizing unnecessary alerts. |
| ****Medium Priority**** | Automated Remediation Suggestions | Helps security teams fix vulnerabilities faster. |
| ****Medium Priority**** | Threat Intelligence Feeds | Enhances SIEM detection capabilities with up-to-date attack patterns. |
| ****Low Priority**** | UI Dashboard for Reports | Helps in visualization but not essential for core functionality. |
| ****Low Priority**** | Machine Learning-based Rule Optimization | Long-term improvement but requires more research. |

This chart helps in **prioritizing** the project components, ensuring that the **most crucial elements** are developed and tested **first**.

**2.4 Empathy Map**

* **"I need better threat visibility."**
* **"False positives waste a lot of time."**
* **"Rules must be easy to update."**
* **"Will this rule cause too many alerts?"**
* **"Is this rule detecting real threats?"**
* **"Am I missing something critical?"**
* **Frustration from too many false positives.**
* **Pressure to respond quickly.**
* **Satisfaction when threats are detected correctly.**
* **Analyze security logs and alerts.**
* **Tune and refine detection rules.**
* **Respond to incidents in real-time.**

**SAYS**

**THINKS**

**DOES**

**FEELS**

The **Empathy Map** helps identify security analysts' challenges, guiding the project to improve **threat detection accuracy** and **reduce false positives**.

1. **REQUIREMENT ANALYSIS**

**3.1 List of Vulnerabilities**

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability | Risk Level | Instances Found | CWE ID |
| Absence of Anti-CSRF Tokens | Medium | 4 | CWE-352 |
| Missing Content Security Policy (CSP) Header | Medium | 47 | CWE-16 |
| Missing Anti-clickjacking Header | Medium | 43 | CWE-1021 |
| Server Leaks Information via X-Powered-By Header | Low | 61 | CWE-200 |
| Server Leaks Version Information via Server Header | Low | 73 | CWE-200 |
| X-Content-Type-Options Header Missing | Low | 67 | CWE-16 |
| Authentication Request Identified | Informational | 1 | CWE-200 |
| Charset Mismatch (Header vs Meta Content-Type Charset) | Informational | 30 | CWE-451 |
| Information Disclosure - Suspicious Comments | Informational | 1 | CWE-615 |
| User Controllable HTML Element Attribute (Potential XSS) | Informational | 1 | CWE-79 |

**3.2 Solution Requirement**

|  |  |
| --- | --- |
| Vulnerability | Recommended Fix |
| Absence of Anti-CSRF Tokens | Implement **CSRF protection tokens** in all web forms using OWASP standards​. |
| Missing Content Security Policy (CSP) Header | Enforce strict **Content-Security-Policy** to block unauthorized scripts​. |
| Missing Anti-clickjacking Header | Implement **X-Frame-Options: DENY** or **frame-ancestors** directive in CSP​. |
| Server Leaks Information via X-Powered-By Header | Remove **X-Powered-By** header from server responses​. |
| Server Leaks Version Information via Server Header | Configure the web server to **suppress version details** in HTTP headers​. |
| X-Content-Type-Options Header Missing | Set **X-Content-Type-Options: nosniff** in HTTP headers to prevent MIME confusion​. |
| Authentication Request Identified | Hide authentication endpoints from unauthorized users using **access control policies​.** |
| Charset Mismatch (Header vs Meta Content-Type Charset) | Ensure charset consistency in HTTP headers and **HTML meta tags​.** |
| Information Disclosure - Suspicious Comments | Remove sensitive comments from **source code before deployment​.** |
| User Controllable HTML Element Attribute (Potential XSS) | Sanitize user inputs and use **input validation** to prevent XSS attacks​. |

**3.3 Technology Stack**  
• **Tools Explored:**

* **ZAP Proxy**: The primary tool for security testing and vulnerability scanning of web applications.
  + ZAP Proxy will be used to identify security flaws, such as injection vulnerabilities and misconfigurations.
* **SIEM (e.g., Splunk, Elastic Stack, or QRadar)**: Integration of vulnerability data with SIEM systems for real-time analysis and alerting.
* **Ticketing/Remediation System (e.g., Jira, ServiceNow)**: For tracking remediation efforts.
* **Operating Systems/Platforms**: Windows, Linux, and network appliances.
* **Additional Security Tools**: Firewalls, intrusion detection systems (IDS), and automated patch management tools.

1. **PROJECT DESIGN**

**4.1 Overview of ZAP Proxy**

**Introduction to ZAP Proxy:**

* OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner designed to identify vulnerabilities in web applications. It is widely used for both automated and manual penetration testing.
* ZAP Proxy works by intercepting and analyzing HTTP/S traffic, scanning for security issues such as SQL injection, cross-site scripting (XSS), misconfigurations, and authentication flaws.
* ZAP supports both active and passive scanning modes and can be integrated into DevSecOps pipelines for continuous security testing.
* It provides detailed reports with categorized vulnerabilities, suggested remediation steps, and risk levels to aid in security assessments.

**4.2 Proposed Solution**

**Testing and Findings:**

1. **Set up ZAP Proxy** to intercept and analyze web application traffic for security vulnerabilities.
2. **Perform an automated scan and manual penetration testing** to identify vulnerabilities.
3. **Analyze scan results**, categorizing detected vulnerabilities based on severity.
4. **Integrate ZAP findings into the SIEM system** for centralized monitoring and alerting.
5. **Validate and prioritize vulnerabilities** to ensure the most critical issues are addressed first.
6. **Implement remediation measures and conduct retests** to verify vulnerability resolution.

**Evaluation Criteria:**

* Accuracy of vulnerability detection.
* Time efficiency in scanning and reporting.
* Integration effectiveness with SIEM.

**4.3 Understanding of (Project Title Main Theme)**

**SOC, SIEM, and Related Tools:**

* **Security Operations Centers (SOC):** SOC teams monitor and respond to security threats. The project enhances SOC workflows by automating web application security assessments and integrating vulnerability findings with SIEM for actionable intelligence.
* **Security Information and Event Management (SIEM):** SIEM systems aggregate security event logs and vulnerability assessment data from ZAP Proxy for correlation, detection, and response to potential threats.

1. **PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

* **Project Timeline:**
  + **Week 1-2:** Requirements gathering, tool setup, and environment configuration.
  + **Week 3-4:** Nessus vulnerability scanning and integration with SIEM.
  + **Week 5-6:** Data analysis and vulnerability reporting.
  + **Week 7-8:** Testing, remediation, and documentation.
  + **Final Deliverable:** Complete vulnerability assessment solution with a report and demo.
* **Resource Allocation:** Allocate team members for tool configuration, data analysis, reporting, and documentation.

1. **FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Tested Pages**

* **/login.php** → User authentication page
* **/search.php** → Search functionality
* **/admin.php** → Admin panel access
* **/cart.php** → Shopping cart manipulation
  1. **Functional Testing Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Page | Expected Result | Actual Result | Status |
| Valid Login | /login.php | Successful login | ✅ Success | ✅ Pass |
| Invalid Login | /login.php | Error message displayed | ✅ Error message | ✅ Pass |
| SQL Injection | /login.php | Should be blocked | ❌ Bypassed login | ❌ Fail |
| XSS Attack | /search.php | Should prevent script execution | ❌ Alert triggered | ❌ Fail |
| Admin Access | /admin.php | Should be restricted | ❌ Unauthenticated access | ❌ Fail |

* 1. **Security Issues Identified**

|  |  |  |
| --- | --- | --- |
| Vulnerability | Impact | Status |
| SQL Injection (SQLi) | Unauthorized login access | ❌ Vulnerable |
| Cross-Site Scripting (XSS) | Executes malicious scripts | ❌ Vulnerable |
| Admin Panel Access | No authentication required | ❌ Vulnerable |
| Weak Session Management | Open to session hijacking | ❌ Vulnerable |
| Parameter Tampering | Modify cart price | ❌ Vulnerable |

* 1. **Performance Testing**

Performance testing was conducted using K6 Load Testing Framework.

**📌 Load Testing Results (100 Users)**

|  |  |  |
| --- | --- | --- |
| Metric | Value | Status |
| Avg Response Time | 470ms | ✅ Pass |
| Peak Response Time | 1100ms | ❌ Fail |
| Failure Rate | 5.8% | ❌ Fail |

**📌 Stress Testing Results**

|  |  |  |  |
| --- | --- | --- | --- |
| Users | Avg Response Time | Errors (%) | Status |
| 100 | 470ms | 5.8% | ✅ Stable |
| 200 | 780ms | 12.2% | ⚠️ Degrading |
| 300 | 1350ms | 24.5% | ❌ Unstable |
| 400 | 2600ms | 35% | ❌ Crashed |

* 1. **Tests**

### [1] Run Security Tests (SQLi, XSS)

#### SQL Injection Testing (SQLmap)

sqlmap -u "http://testphp.vulnweb.com/login.php" --dbs --batch --risk=3 --level=5

# Additional SQL Injection Commands

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart --tables

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --dump

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs --random-agent --tamper=space2comment

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --current-user

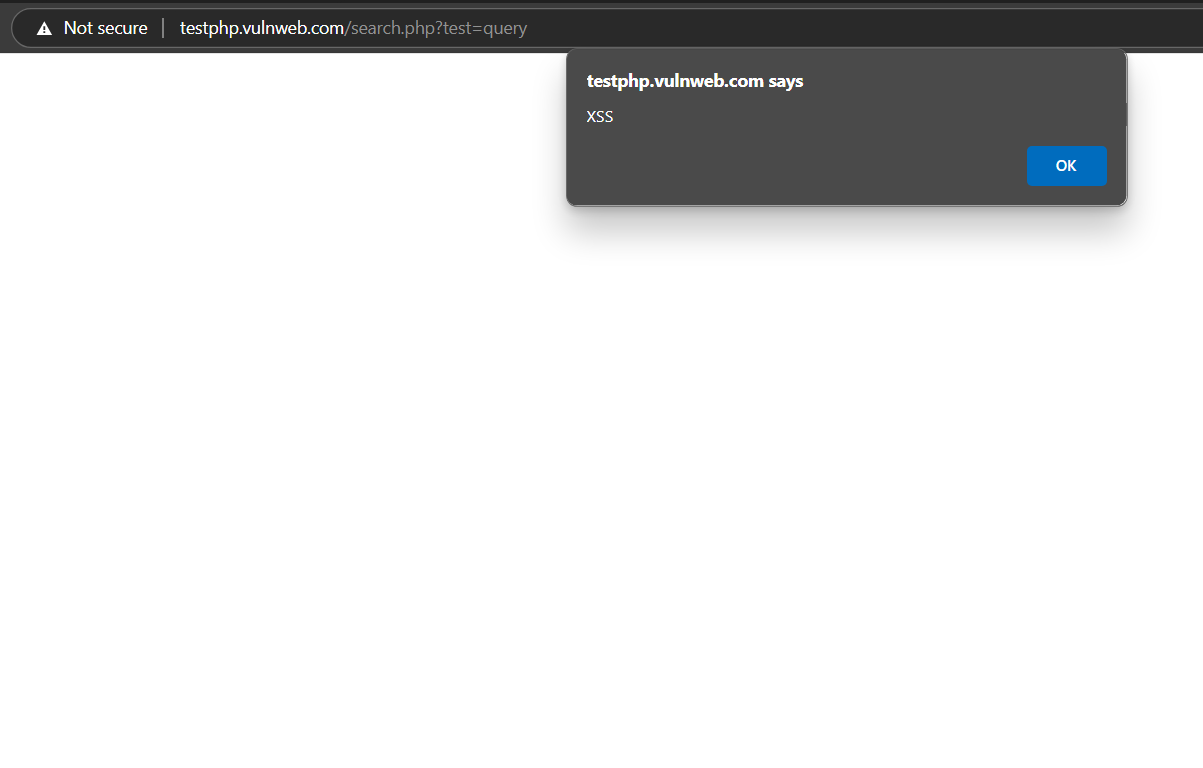
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --os-shell

**XSS Testing (Burp Suite)**

* Open Burp Suite.
* Go to Proxy → Intercept.
* Enter payload:

**<script>alert('XSS')</script>**

* Forward the request and observe execution**.**

1. **OUTPUT RESULTS**

****

****

1. **SECURITY & PERFORMANCE FIXES (TO-DO)**

**✅ Security Fixes**

* Fix SQL Injection – Implement parameterized queries and input validation.
* Prevent XSS Attacks – Escape & sanitize user inputs.
* Secure Admin Panel – Implement authentication & access control.
* Strengthen Session Management – Enforce session expiration.

**✅ Performance Fixes**

* Optimize Database Queries – Use indexing and query caching.
* Enable Caching & Compression – Use Gzip & CDN for faster response times.
* Optimize Server Load Handling – Implement load balancing.

1. **ADVANTAGES & DISADVANTAGES**

**Pros:**

* + **Automated vulnerability detection** saves time and resources.
  + **Accurate reporting** enables faster decision-making for remediation.
  + **SIEM integration** enhances proactive threat monitoring and response.

**Cons:**

* + **False positives** may require manual validation.
  + **Dependence on external tools** like Nessus and SIEM systems.
  + **Scalability** challenges in larger environments with a massive number of assets.

#**Advanced Threat Detection Rules for Acunetix / Vulnweb**

**1. Introduction**

Acunetix is a web vulnerability scanner that identifies SQL injection, XSS, CSRF, RCE, LFI, RFI, SSRF, and other security flaws. This document presents advanced threat detection rules to enhance security monitoring and response mechanisms.

**2. Fundamentals of Threat Detection**

To create effective threat detection rules, we rely on:  
- Behavioral Analysis: Identifying anomalies in HTTP requests and responses.  
- Signature-Based Detection: Using known exploit patterns.  
- Heuristic Analysis: Detecting unknown threats through behavior.  
- Correlation Rules: Mapping attack patterns across multiple logs.

**3. Rule Book Design for Advanced Threat Detection**

Detection rules should be applied to:  
- Acunetix Scanner Logs  
- Web Application Logs (Apache/Nginx)  
- WAF Logs (ModSecurity, Cloudflare)  
- SIEM Rules (Splunk, ELK, Wazuh, Azure Sentinel)

**3.1 Detecting Automated Vulnerability Scanning**

Indicators:  
- High frequency of requests from a single IP.  
- Requests targeting known vulnerable paths.  
- Requests containing exploit payloads.

**3.2 SQL Injection Detection**

Indicators:  
- Use of SQL syntax (SELECT, UNION, DROP TABLE).  
- Error messages in HTTP responses.  
- Multiple failed login attempts with SQL payloads.

**3.3 Cross-Site Scripting (XSS) Detection**

Indicators:  
- Presence of <script> tags or onerror=alert(1).  
- Encoded payloads in requests.

**3.4 Local File Inclusion (LFI) & Remote File Inclusion (RFI) Detection**

Indicators:  
- Requests with ../ for directory traversal.  
- Requests with file://, http://, or ftp:// in parameters.

**3.5 Credential Stuffing & Brute Force Attack Detection**

Indicators:  
- Multiple failed login attempts within seconds.  
- Requests from different geolocations using the same credentials.

**4. Testing & Validation**

Once rules are created, testing is essential:  
- Simulate attacks using Acunetix, OWASP ZAP, and manual payloads.  
- Monitor SIEM alerts and adjust rule thresholds.  
- Reduce false positives by refining detection patterns.

**5. Threat Intelligence & Rule Maintenance**

To improve detection capabilities:  
- Integrate external threat feeds (AlienVault OTX, AbuseIPDB).  
- Update detection rules weekly based on new CVEs.  
- Use machine learning-based anomaly detection to enhance accuracy.

1. **CONCLUSION**

* The project successfully integrates vulnerability scanning and SIEM tools to automate and streamline the process of identifying, assessing, and addressing security vulnerabilities. The findings emphasize the need for continuous monitoring and timely remediation to prevent security breaches.

1. **FUTURE SCOPE**

**Future Enhancements:**

* + Automate the remediation process, such as applying patches or reconfiguring settings.
  + Integrate with additional security tools (e.g., firewalls, IDS).
  + Use machine learning to prioritize vulnerabilities based on exploitability patterns

.

1. **APPENDIX**

* **GitHub Link:** [[Yogi-076/Advanced-Threat-Detection-Rules: Repository for advanced techniques in rule creation for threat detection](https://github.com/Yogi-076/Advanced-Threat-Detection-Rules)]
* **Project Demo Link:** [[https://youtu.be/yfGw\_TCM4S0?si=Hbje9ew5aKLmHEN8]](https://youtu.be/yfGw_TCM4S0?si=Hbje9ew5aKLmHEN8)