International Journal on Optimization and Applications
IJOA. Vol. 05, Issue No. 03, Year 2025, www.ijoa-journal.com
Copyright ©2025 by **I**nternational **J**ournal on **O**ptimization and **A**pplications

Automating the Pentest Lifecycle: From Recon to Reporting in a PTES-Compliant PTaaS Platform

1st Mohamed Ait Ouaarab
ENSA, Ibn Tofail University
mohamed.aitouaarab@uit.ac.ma

2nd Dr. Hanaa Hachimi
ENSA, Ibn Tofail University
hanaa.hachimi@uit.ac.ma

3rd Abdelillah Salhi Senhaji
Valinnov
abdelillah.ssenhaji@valinnov.tech

***Abstract*—The escalating sophistication of cyber threats exposes the inherent scalability and resource limitations of traditional manual penetration testing. This paper introduces PTaaS (Penetration Testing as a Service), an innovative, autonomous platform designed to revolutionize web application security assessment and management through an intelligent, playbook-driven methodology. The platform's architecture is meticulously engineered to automate 60% of the penetration testing lifecycle, strictly adhering to the Penetration Testing Execution Standard (PTES) for systematic and repeatable assessments. PTaaS integrates a diverse array of reconnaissance, vulnerability analysis, and exploitation capabilities. These functions are unified by a central wrapper, which serves as an intelligent intermediary between the platform's core engine and its various modular scripts. A cornerstone of the platform is its adaptive playbook management system, which codifies expert knowledge into dynamic, multi-stage attack workflows. This allows the platform to simulate real-world adversary tactics with unprecedented precision, moving beyond isolated vulnerability detection to demonstrate complex attack chains. Furthermore, an integrated AI-driven engine provides contextual risk assessment and intelligent remediation guidance, transforming raw vulnerability data into prioritized, actionable insights. By democratizing access to high-fidelity security assessments, PTaaS empowers organizations to achieve continuous security validation and gain a deeper, more accurate understanding of their true security posture.**

***Keywords*—Automation, Penetration Testing, Management, Cybersecurity, PTES**

I. Introduction

In an era of escalating cyber threats and increasingly sophisticated attack vectors, penetration testing has become a cornerstone of a robust cybersecurity strategy. It provides a proactive approach to identifying and mitigating vulnerabilities by simulating real-world attacks on an organization's digital infrastructure. However, traditional manual penetration testing, while thorough, is beset by significant limitations. It is a time-consuming, costly, and resource-intensive process, heavily dependent on the availability of a limited pool of highly skilled security experts. This manual approach struggles to keep pace with the dynamic nature of modern IT environments and the rapid deployment cycles of new software and infrastructure.

To address these challenges, the cybersecurity industry is increasingly turning to pentest automation. Automated penetration testing leverages software tools to simulate cyberattacks, enabling organizations to conduct security assessments with greater speed, efficiency, and consistency. This automation allows for more frequent and scalable testing across large and complex networks, including cloud and IoT environments. The integration of artificial intelligence (AI) and machine learning (ML) is further enhancing these capabilities, enabling more adaptive and intelligent threat detection and analysis. These technologies can significantly reduce the time required to identify vulnerabilities, analyze large datasets, and even predict potential attack paths.

However, the adoption of automated penetration testing introduces new management complexities. Organizations must effectively manage the automated testing process, from tool selection and configuration to the interpretation and prioritization of results. A significant challenge lies in handling the volume of data generated by automated tools and filtering out false positives to ensure that remediation efforts are focused on genuine threats. Furthermore, integrating the findings of automated penetration testing into the broader security and software development lifecycle (SDLC) is crucial for timely vulnerability remediation and a holistic security posture. This requires clear communication and collaboration between security, IT operations, and development teams.

This research article explores the critical intersection of pentest automation and management. It examines the benefits and challenges of automated testing, analyzes current trends and technologies, and proposes a framework for the effective management of automated penetration testing programs. By addressing the operational and strategic aspects of this evolving field, this paper aims to provide valuable insights for organizations seeking to enhance their security posture in an increasingly hostile digital landscape. [1] [2]

II. Related Work

The landscape of automated security testing has evolved significantly, moving from isolated scanners to more integrated and intelligent platforms. The PTaaS platform described in this paper builds upon concepts from several established domains, including traditional vulnerability scanning, Penetration Testing as a Service (PTaaS), Breach and Attack Simulation (BAS), and Security Orchestration, Automation, and Response (SOAR).

A. Traditional and Advanced Vulnerability Scanners

Foundational tools for security assessment include network and web application scanners like Nmap, Nessus, OpenVAS, and OWASP ZAP. These tools are highly effective at identifying known vulnerabilities and misconfigurations based on signatures. More advanced solutions, such as Burp Suite Professional, provide extensive capabilities for manual and semi-automated web application testing. While our PTaaS platform integrates the functions of such tools, it differentiates itself by orchestrating them as part of a broader, objective-driven workflow rather than executing them as standalone scans. [3] [4] [5]

B. Penetration Testing as a Service (PTaaS) Platforms

The PTaaS model has been commercialized by companies like Synack, Cobalt, and Bugcrowd. These platforms typically combine automated scanning with a community of human penetration testers to provide continuous security assessments. They excel at leveraging human intelligence for discovering complex and logic-based vulnerabilities. However, our proposed platform focuses on maximizing the potential of full autonomy, using AI and dynamic playbooks to simulate the cognitive processes of a human tester, thereby offering a highly scalable and repeatable alternative. [6] [7]

C. Integrated and AI-Driven Security Platforms

BAS platforms, such as those offered by Picus Security, Mandiant, and Cymulate, are designed to continuously test an organization's security controls against real-world attack scenarios. They validate the effectiveness of firewalls, endpoint detection, and other defensive measures. While BAS platforms are excellent at simulating adversary tactics and validating controls, our PTaaS platform extends this concept by focusing on the discovery of unknown vulnerabilities and demonstrating their exploitability within the context of the full PTES lifecycle, from reconnaissance to post-exploitation.

D. Security Orchestration, Automation, and Response

SOAR platforms like Splunk SOAR, Palo Alto Networks Cortex XSOAR, and IBM Security QRadar SOAR provide the architectural blueprint for the orchestration capabilities of our PTaaS solution. SOAR tools are designed to integrate disparate security solutions and automate workflows, often for incident response. Our platform adopts this orchestration philosophy but customizes it for proactive, offensive security operations. The "wrapper" component in our architecture functions similarly to a SOAR engine, acting as the central nervous system that intelligently coordinates the execution of various scripts and modules to achieve a specific testing objective. [8]

E. Integrated and AI-Driven Security Platforms

Emerging platforms are increasingly incorporating Artificial Intelligence to enhance their analytical capabilities. For instance, CrowdStrike utilizes AI for threat detection and endpoint protection, and platforms like Tenable use machine learning to predict which vulnerabilities are most likely to be exploited. Our PTaaS platform aligns with this trend but focuses the application of AI on contextual risk assessment and generating tailored remediation guidance based on demonstrated attack paths, moving beyond probabilistic analysis to provide deterministic security intelligence.

In summary, while elements of our proposed PTaaS platform exist across the cybersecurity ecosystem, its unique contribution lies in the holistic integration of these capabilities. By combining a PTES-aligned methodology, a central orchestration wrapper, a dynamic playbook system for multi-stage attacks, and an AI engine for contextual analysis, the platform aims to create a truly autonomous system that bridges the gap between vulnerability scanning, breach simulation, and human-led penetration testing.

III. Architecture and Methodology

The PTaaS architecture is designed as a sequential, multi-phase workflow, orchestrated by a central controller that manages the flow of data between specialized analysis and action modules. The entire process is engineered to transform raw reconnaissance data into actionable, context-aware security intelligence.

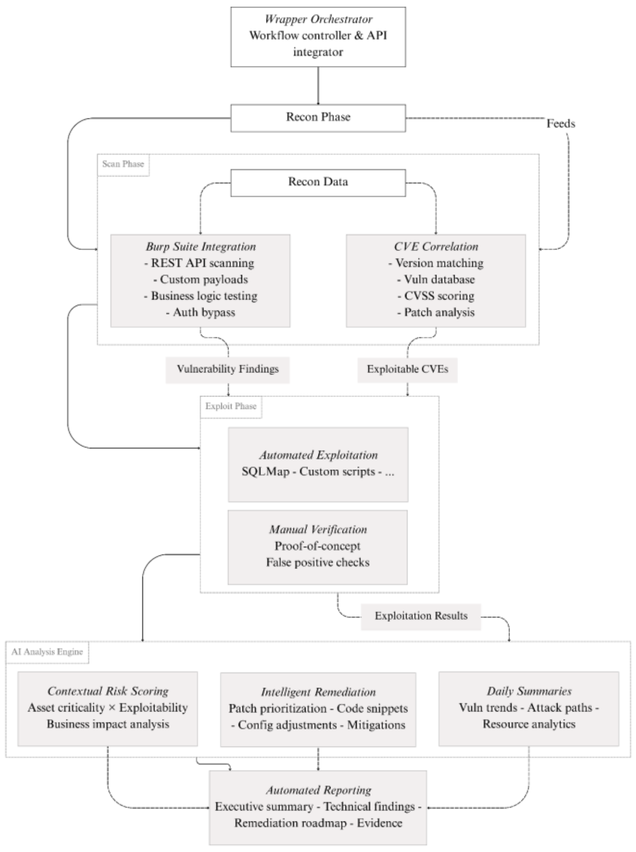

Fig. 1. PTaaS High-Level Architecture

A. *Wrapper Orchestrator: The Central Control Plane*

At the highest level, the Wrapper Orchestrator functions as the system's central nervous system. It is a workflow controller and API integrator responsible for initiating the entire penetration testing process, managing the sequence of operations, and ensuring the seamless passage of data between distinct phases. It receives the initial target information and triggers the first phase of the assessment

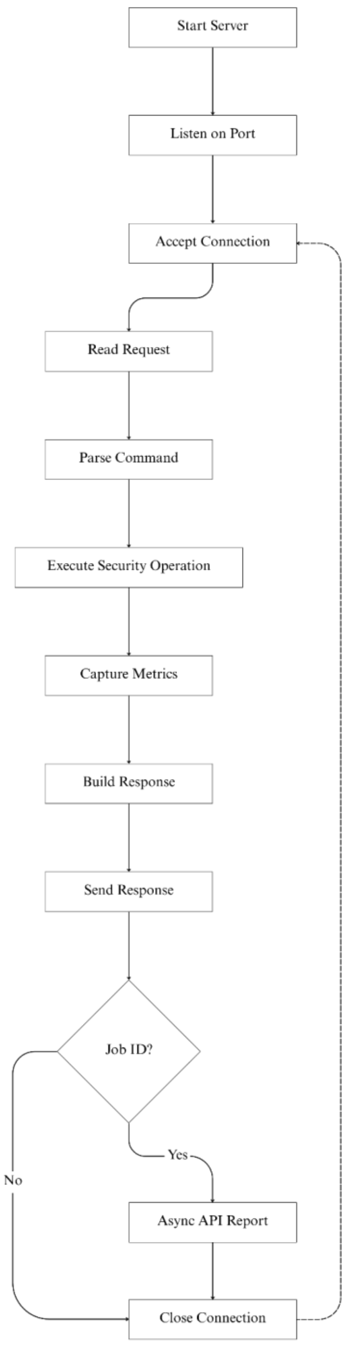

Fig. 2. Wrapper Orchestrator Workflow

B. *Phase 1: Reconnaissance*

The Recon Phase is the initial data acquisition stage. The orchestrator invokes a suite of passive and active reconnaissance modules to gather foundational intelligence about the target environment. This includes, but is not limited to, DNS enumeration, subdomain discovery, port scanning, and technology fingerprinting (e.g., identifying web servers, frameworks, and backend languages). The output of this phase is a structured Recon Data object, typically in JSON format, containing a comprehensive profile of the target's attack surface. This data is the critical input for the subsequent Scan Phase.

C. *Phase 2: Scanning and Vulnerability Identification*

The Scan Phase takes the Recon Data as input and performs deep analysis to identify potential security weaknesses. This phase operates on two parallel tracks to ensure comprehensive coverage:

1) *Burp Suite Integration:*

This track focuses on dynamic application security testing (DAST). Leveraging Burp Suite's REST API (Montoya), the orchestrator programmatically launches highly targeted scans. This integration allows for sophisticated testing that goes beyond standard signatures, including:

Custom Payloads: Injecting context-specific payloads to test for vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), and Server-Side Request Forgery (SSRF).

Business Logic Testing: Executing sequences of requests to identify flaws in application logic that standard scanners would miss.

Authentication Bypass: Probing for weaknesses in authentication and session management mechanisms.

The output of this track is a set of Vulnerability Findings, detailing potential flaws discovered through active interaction.

2) *CVE Correlation:*

This track performs automated vulnerability discovery based on the fingerprinted technologies from the Recon Data. It involves:

Version Matching: Comparing the versions of identified software (e.g., WordPress 5.8, Apache 2.4.52) against a comprehensive vulnerability database.

Business Logic Testing: Executing sequences of requests to identify flaws in application logic that standard scanners would miss.

Authentication Bypass: Probing for weaknesses in authentication and session management mechanisms.

The output of this track is a set of Vulnerability Findings, detailing potential flaws discovered through active interaction.

D. *Phase 3: Exploitation and Verification*

The Exploit Phase is designed to validate the findings from the Scan Phase and demonstrate their real-world impact. It ingests both the Vulnerability Findings and Exploitable CVEs to perform controlled exploitation attempts.

- *Automated Exploitation:* This sub-module uses specialized tools and scripts [9] (e.g., SQLMap for database exploitation, Metasploit modules, or custom Python scripts) to actively try and leverage the identified vulnerabilities. The goal is to confirm exploitability and, if successful, perform actions like data exfiltration or privilege escalation.

- *Manual Verification:* This is a crucial quality assurance step to eliminate false positives. The system generates a Proof-of-Concept (PoC) for each successful exploit and runs additional checks to ensure the finding is genuine. This may involve a semi-automated workflow where high-impact findings are flagged for human review.

The final output is a high-fidelity list of Exploitation Results, containing only confirmed, verifiable vulnerabilities with clear evidence of their impact.

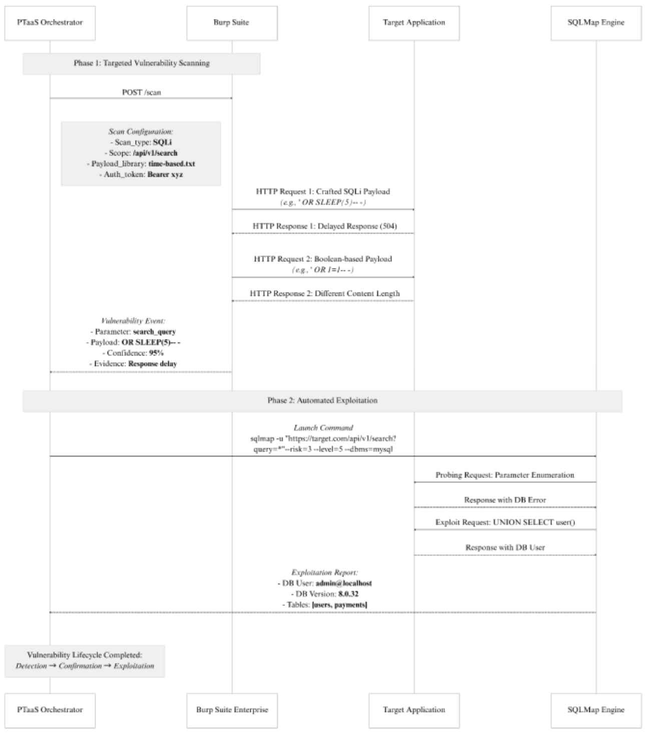
PTaaS Orchestrator
Burp Suite
Target Application
SQLMap Engine
Phase 1: Targeted Vulnerability Scanning
POST /scan
Scan Configuration:
- Scan_type: SQLi
- Scope: /api/v1/search
- Payload_library: time-based.txt
- Auth_token: Bearer xyz
HTTP Request 1: Crafted SQLi Payload (e.g., ' OR SLEEP(5)-- -)
HTTP Response 1: Delayed Response (504)
HTTP Request 2: Boolean-based Payload (e.g., ' OR 1=1-- -)
HTTP Response 2: Different Content Length
Vulnerability Event:
- Parameter: search_query
- Payload: OR SLEEP(5)-- -
- Confidence: 95%
- Evidence: Response delay
Phase 2: Automated Exploitation
Launch Command
sqlmap -u "https://target.com/api/v1/search?query=*"--risk=3 --level=5 --dbms=mysql
Probing Request: Parameter Enumeration
Response with DB Error
Exploit Request: UNION SELECT user()
Response with DB User
Exploitation Report:
- DB User: admin@localhost
- DB Version: 8.0.32
- Tables: [users, payments]
Vulnerability Lifecycle Completed:
Detection → Confirmation → Exploitation
PTaaS Orchestrator
Burp Suite Enterprise
Target Application
SQLMap Engine

Fig. 3. Exploitation and Verification Process

Figure 4 illustrates the internal flow of the SQL injection playbook as implemented in the Exploitation and Verification phase. When reconnaissance or scanning results indicate a potential SQL injection vulnerability, the Wrapper Orchestrator triggers the relevant playbook with the necessary parameters. The playbook automates payload generation, injection, and response analysis, detecting database-specific signatures or anomalies. Upon confirmation, it proceeds to exploitation steps such as controlled data exfiltration. The process incorporates error handling for blocked attempts or inconclusive results, and integrates verification to minimize false positives. This modular approach ensures that exploitation remains both targeted and reproducible within the PTES-aligned workflow.

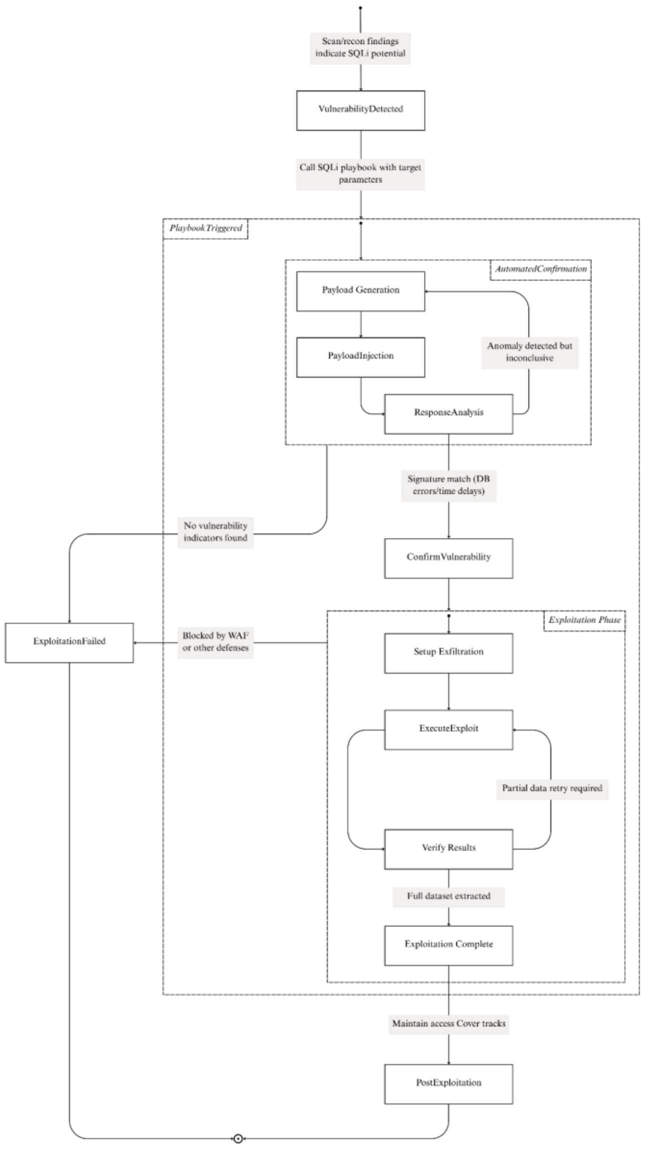
Scan/recon findings indicate SQLi potential
VulnerabilityDetected
Call SQLi playbook with target parameters
PlaybookTriggered
AutomatedConfirmation
Payload Generation
PayloadInjection
Anomaly detected but inconclusive
ResponseAnalysis
Signature match (DB errors/time delays)
No vulnerability indicators found
ConfirmVulnerability
ExploitationFailed
Blocked by WAF or other defenses
Exploitation Phase
Setup Exfiltration
ExecuteExploit
Partial data retry required
Verify Results
Full dataset extracted
Exploitation Complete
Maintain access Cover tracks
PostExploitation

Fig. 4. SQL Injection Playbook Execution Flow

E. *Phase 4: AI-Driven Analysis and Reporting*

The confirmed Exploitation Results are fed into the AI Analysis Engine, which transforms technical data into strategic intelligence. [1] [2]

- *Contextual Risk Scoring:* The engine moves beyond generic CVSS scores by calculating a contextual risk. It uses a model that considers Asset Criticality (Is this a production database or a dev server?), technical Exploitability (How easy is it to leverage this flaw?), and potential Business Impact (What is the financial or reputational damage of a breach?).

- *Intelligent Remediation:* Based on the contextual risk, the engine generates actionable remediation guidance. This includes Patch Prioritization (telling teams what to fix first), providing secure Code Snippets, and suggesting specific Configuration Adjustments or mitigation strategies.

- *Daily Summaries:* For ongoing assessments, the engine provides trend analysis, visualizes common Attack Paths, and offers analytics on resource utilization.

Finally, all this processed intelligence is compiled into a comprehensive Automated Report, featuring a high-level Executive Summary for management, detailed Technical Findings and Evidence for security teams, and a prioritized Remediation Roadmap for developers

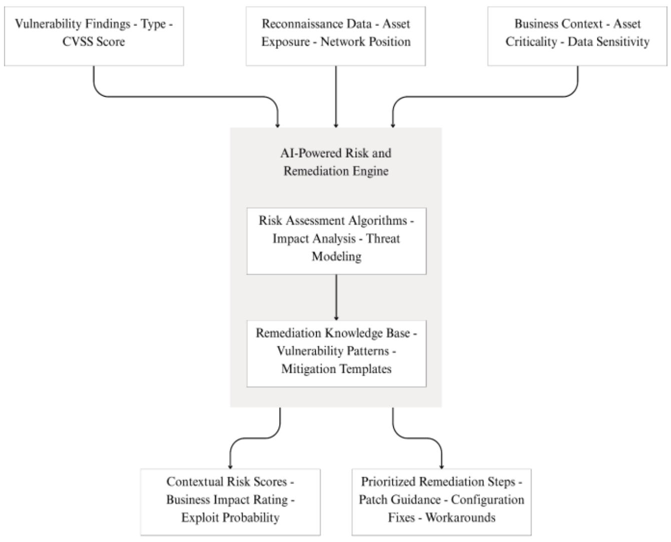
Vulnerability Findings - Type - CVSS Score
Reconnaissance Data - Asset Exposure - Network Position
Business Context - Asset Criticality - Data Sensitivity
AI-Powered Risk and Remediation Engine
Risk Assessment Algorithms - Impact Analysis - Threat Modeling
Remediation Knowledge Base - Vulnerability Patterns - Mitigation Templates
Contextual Risk Scores - Business Impact Rating - Exploit Probability
Prioritized Remediation Steps - Patch Guidance - Configuration Fixes - Workarounds

Fig. 5. AI-Driven Risk Analysis and Reporting

IV. IMPLEMENTATION

A. *Reconnaissance Engine*

The PTaaS platform's reconnaissance and analysis capabilities are implemented through specialized modules covering the full spectrum of information gathering and vulnerability identification. The process progresses from passive intelligence collection to active probing and automated vulnerability correlation.

Passive reconnaissance begins with the Web Archive Analysis module, which leverages public archives (e.g., Wayback Machine) to uncover historical endpoints, subdomains, and sensitive data patterns, and the DNS Reconnaissance module, which extracts intelligence from public DNS records.

Active reconnaissance is then performed by the Active Subdomain and Directory Detection module, using DNS bruteforcing and high-speed fuzzing to map the accessible attack surface.

Technology and configuration assessment follows through the WAF and Technology Stack Detection module, the Security Header and Cookie Analysis module, and the SSL/TLS Configuration Verification service, which collectively profile security technologies, evaluate protocol configurations, and detect misconfigurations.

Finally, the Vulnerability and CVE Check Engine correlates collected data (e.g., software versions, services) against vulnerability databases such as NVD and MITRE CVE, producing an actionable list of known security flaws for exploitation and reporting.

B. *Vulnerability Scanning Engine*

Following reconnaissance, the PTaaS platform advances to direct vulnerability discovery via its Vulnerability Scanning Engine, a dedicated microservice integrating Burp Suite Enterprise Edition through its REST API. Unlike generic scanners, this engine performs context-aware, targeted scans informed by the comprehensive intelligence gathered during reconnaissance.

The process begins when the Wrapper Orchestrator issues a directive containing the full structured JSON output from prior modules, including discovered endpoints, technologies, versions, and credentials. The engine uses this data to dynamically generate a custom scan configuration, defining scope, authentication settings, and technology-specific checks to optimize accuracy and performance.

Once configured, the engine initiates a Burp scan via the /scan endpoint, performing intelligent crawling followed by an in-depth audit for vulnerabilities such as SQL injection, XSS, and SSRF. During execution, it continuously polls the /scan/{scan_id} endpoint to monitor progress and stream live findings to the user interface.

Upon completion, all results are normalized into the platform's standardized JSON schema, ensuring consistency and enabling seamless integration with subsequent exploitation, verification, and AI-assisted analysis phases.

C. *Exploitation Engine*

After vulnerability detection, the PTaaS platform enters the Exploitation and Verification phase, focusing on controlled validation rather than indiscriminate exploitation. Its goal is to confirm exploitability, remove false positives, and demonstrate tangible impact.

This phase operates as a set of containerized, on-demand services—each dedicated to a specific vulnerability type and orchestrated via the Wrapper Orchestrator. Normalized scan results are consumed, and for each supported vulnerability, the orchestrator dispatches tasks to the appropriate service. For example, SQL injection findings are validated using a dedicated SQLMap service to confirm the flaw and safely enumerate limited database details. Commix is employed for OS command injection, while XSStrike validates XSS through contextual payload generation and headless browser proof.

Where automated tools fall short, the framework supports custom exploitation scripts for complex logic flaws or chained exploits, ensuring a hybrid automated-manual capability. All actions are performed under strict non-destructive parameters to maintain target stability.

Each confirmed vulnerability is returned as a structured JSON object containing payloads, responses, and proof artifacts (e.g., screenshots, exfiltrated sample data). This validated dataset is published to the message bus for use in AI-driven risk analysis and final reporting.

V. CASE STUDY

Table 1 presents a comparative overview of traditional penetration testing and Penetration Testing as a Service

(PTaaS). The comparison focuses on four key dimensions—delivery model, frequency, communication, and reporting—to highlight the operational and strategic differences between the two approaches. While traditional penetration testing is typically project-based with limited communication channels and static reports, PTaaS offers a continuous, on-demand testing model supported by integrated collaboration platforms and real-time reporting. This shift addresses the need for faster vulnerability identification, streamlined remediation workflows, and greater adaptability in modern threat landscapes. [4] [5]

Table 1. Comparative Overview

	Manual Penetration Testing	**Existing Automated Tools**	**Proposed PTaaS Platform**
Time Efficiency	Low; requires extensive manual effort	High; rapid scanning but limited adaptability	Improved; automation accelerates routine tasks
Contextual Awareness	High; deep understanding of target environment	Low; generic scanning without customization	High; dynamic scan configurations based on reconnaissance
False Positive Rate	Low; results validated by expert analysis	High; frequent false positives	Reduced; targeted scanning combined with verification
Authentication and Session Handling	Manual and tailored	Often limited or static	Automated and adaptive, enabling deep authenticated scanning
Vulnerability Verification	Manual exploitation and confirmation	Minimal or absent verification	Automated verification using specialized tools and scripts
Detection of Complex Logic Flaws	Effective through human insight	Poor capability	Hybrid approach combining automated and custom scripted tests
Scalability and Repeatability	Limited by human resources	High scalability	Highly scalable due to containerized microservices
Required Expertise	High expertise required	Low to moderate expertise	Moderate; system facilitates semi-automation with expert oversight

Figure 6 illustrates the distribution of AI-generated security recommendations within the developed penetration testing automation platform. The analysis categorizes these recommendations into three distinct outcomes: accepted, modified, and rejected. The quantitative breakdown reveals that 78% of the AI-generated suggestions were accepted without modification, indicating a high level of precision and contextual relevance in the automated assessment process. This outcome suggests that the AI module is effectively aligned with the PTES-based workflow and capable of producing remediation steps that require minimal human intervention.

A further 15% of the AI-generated suggestions were modified prior to implementation. This proportion reflects scenarios in which the automated recommendations were directionally correct but required contextual adjustments to align with specific system architectures, organizational policies, or operational constraints. The need for modification is an expected characteristic in semi-automated security workflows, where the interplay between machine-generated outputs and expert human validation enhances the accuracy and applicability of the final remediation steps.

The remaining 7% of AI-generated suggestions were rejected outright. This segment is indicative of either false positives in the detection pipeline or remediation strategies that were deemed non-viable due to technical, operational, or compliance-related limitations. While relatively small, this proportion highlights the importance of maintaining a human-in-the-loop architecture to ensure that automated processes do not introduce misaligned or potentially disruptive security changes.

From a mathematical perspective, if N represents the total number of AI-generated suggestions, the distribution can be expressed as: Accepted = $0.78 \times N$, Modified = $0.15 \times N$, and Rejected = $0.07 \times N$. For instance, with N = 500 suggestions, this corresponds to 390 accepted, 75 modified, and 35 rejected cases. These values provide both a performance indicator of the AI suggestion engine and a benchmark for iterative improvement.

Overall, the observed distribution underscores the efficacy of integrating AI-driven risk detection and remediation guidance into penetration testing processes. The high acceptance rate reflects strong alignment between automated outputs and expert expectations, while the presence of modified and rejected cases justifies the hybrid human-AI validation layer essential in mission-critical cybersecurity operations.

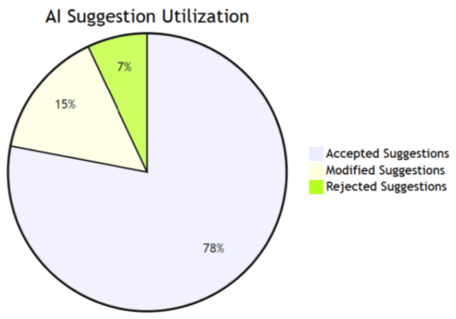

Fig. 6. AI-Generated Security Recommendations: Acceptance Breakdown

The adoption of automation within the PTaaS platform significantly reduces the time required for each penetration testing phase. As shown in Table 2, reconnaissance drops from approximately 10 hours manually to 1.2 hours in a fully automated workflow. Vulnerability scanning experiences a similar reduction, from 15 hours to 1.5 hours, while exploitation time decreases from 8 hours to 1 hour. Even the reporting phase, traditionally time-intensive, is streamlined from 7 hours to 10 min. Overall, the platform achieves substantial efficiency gains, allowing security teams to focus more on analysis and decision-making rather than repetitive manual tasks.

Table 2. Efficiency Metrics

Phase	**Manual**	**Semi-Auto**	**Full Auto**
Reconnaissance	10 hrs	4 hrs	1.2 hrs
Vulnerability Scanning	15 hrs	6 hrs	1.5 hrs
Exploitation	8 hrs	5 hrs	1 hrs
Reporting	7 hrs	5 hrs	10 min

VI. CONCLUSION

This work presents a PTES-aligned PTaaS platform that automates part of the pentesting lifecycle from reconnaissance to reporting. By integrating AI-driven vulnerability analysis, dynamic playbooks for task sequencing, and a central orchestration engine, the platform streamlines and standardizes key test phases. These capabilities significantly accelerate testing workflows and improve consistency, though some steps (such as complex exploitation and validation) still require human expertise. Full end-to-end automation is acknowledged to be a work in progress, and future efforts will focus on expanding automation coverage to the remaining phases. Planned enhancements include enriching the playbooks, incorporating advanced adversarial emulation for more realistic threat scenarios, and further leveraging AI to handle increasingly complex tasks.

REFERENCES

[1] C. Skandylas and M. Asplund, “Automated Penetration Testing: Formalization and Realization,” *arXiv preprint arXiv:2412.12745*, 2024.

[2] W. Zhang, J. Xing, and X. Li, “Penetration Testing for System Security: Methods and Practical Approaches,” *arXiv preprint arXiv:2505.19174*, 2025.

[3] F. A. Saeed, “Using WASSEC to Evaluate Commercial Web Application Security Scanners,” *ResearchGate Publication*, 2014.

[4] S. Alazmi and D. C. De Leon, “A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners,” *IEEE Open Journal of the Computer Society*, vol. 10, 2022.

[5] u. Authors, “An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities,” *Electronics (MDPI)*, vol. 11, no. 19, 2022.

[6] B. Kurniawan, I. Ruslianto, and S. Bahri, “Implementation of Penetration Testing on the Website Using the Penetration Testing Execution Standard (PTES) Method,” *CESS (Journal of Computer Engineering, System and Science)*, vol. 8, no. 2, pp. 518–528, 2023.

[7] S. U. Sunaringtyas and D. S. Prayoga, “Implementasi Penetration Testing Execution Standard untuk Uji Penetrasi pada Layanan Single Sign-On,” *Edu Komputika Journal,* vol. 8, no. 1, 2023.

[8] P. Modesti, L. Golightly, L. Holmes, C. Opara, and M. Moscini, “Bridging the Gap: A Survey and Classification of Research-Informed Ethical Hacking Tools,” *arXiv preprint arXiv:2407.14255*, 2024.

[9] M. Baklizi, I. Atoum, N. Abdullah, O. A. Al-Wesabi, A. A. Otoom, and M. A.-S. Hasan, “A Technical Review of SQL Injection Tools and Methods: A Case Study of SQLMap,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 3, pp. 75–85, 2022.