ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Penetration Testing for Websites using LLM

Dr. Vilas S. Gaikwad[1], Soham Deshmukh[4], Mohanish Kulkarni[2], Pallavi Akolkar[3] , Jayam Mehta[5]

[1]Dept. of Information Technology Trinity College of Engineering and Research

[4]Dept. of Information Technology Trinity College of Engineering and Research

[2]Dept. of Information Technology Trinity College of Engineering and Research

[3,5]Dept. of Information Technology Trinity College of Engineering and Research

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150500244

Received: 27 May 2026; Accepted: 01 June 2026; Published: 22 June 2026

ABSTRACT

Despite growing interest in applying Large Lan-guage Models (LLMs) to security assessment, existing prototype systems rarely provide controlled benchmarks, reproducible ar-tifacts, significance testing, or structured human evaluation. We introduce a reproducible, safety-bounded LLM-augmented web assessment framework that layers an LLM reasoning module atop conventional scanners to perform context-aware triage, con-strained payload proposal, evidence-grounded response analysis, false-alarm suppression, and actionable fix guidance. Empirical evaluation across WebGoat v2023.4, DVWA v1.9, bWAPP v2.2, and five authorized staging replicas of disclosed targets — spanning 1,247 endpoints and 462 labeled ground-truth flaws shows that the proposed approach raises micro-averaged F1 from 0.58 to 0.82 over scanner-only baselines, lowers the false-alarm rate from 23.4% to 7.2% (McNemar $p<0.001$), and shortens analyst review time by 43%, from 47 to 27 minutes per engagement (Wilcoxon $p=0.008$). Over 2,500 test executions, the safety layer blocked every destructive candidate payload with zero unsafe executions. A structured 10-person evaluation with five security practitioners and five application developers con-firmed statistically significant improvements in comprehension speed, report clarity, and remediation actionability ($p<0.01$).

Index Terms: Large Language Models, web penetration test-ing, vulnerability assessment, OWASP, CWE, reproducibility, statistical evaluation

INTRODUCTION

Web applications present a persistently wide attack surface, coupling server-side business logic with client-side scripting, authentication subsystems, REST and GraphQL APIs, file-handling endpoints, database layers, and third-party depen-dencies. Expert-led security assessments remain thorough but expensive and slow; automated scanners extend coverage at speed yet generate findings that are context-free and often noisy. This work investigates whether LLMs can serve as a practical reasoning intermediary — improving triage and reporting without acting as an unconstrained autonomous agent.

The system described here positions deterministic scanning tools as evidence collectors and delegates five bounded cog-nitive tasks to an LLM: prioritizing candidate findings by context, proposing constrained test payloads, interpreting tool responses, suppressing false alarms, and generating structured fix guidance. Evaluation proceeds against labeled ground truth with paired statistical comparison against both a scanner-only baseline and PentestGPT.

Primary contributions include:

1) A fully reproducible LLM-augmented web security assess-ment architecture featuring local model inference, rule-based safety filters, and schema-validated JSON outputs.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

2) A labeled benchmark covering 1,247 endpoints and 462 confirmed vulnerabilities across WebGoat, DVWA, bWAPP, and five authorized real-world staging targets.

3) Per-CWE precision, recall, F1, and false-positive rate (FPR) with McNemar significance testing against two baselines.

4) A component-level ablation study quantifying each LLM module's individual contribution.

5) A structured 10-participant evaluation measuring utility for both security practitioners and developers.

6) Full public release of source code, prompt templates, Docker artifacts, ground-truth labels, raw logs, and analysis scripts.

Related Work

Classical Penetration Testing

Security assessment methodology has long been structured around phased workflows: information gathering, weakness discovery, exploitation confirmation, evidence documentation, and remediation tracking. Bacudio et al. [1] formalize this process and highlight the centrality of repeatable evidence collection. The principal limitation is scalability: human-led assessments cannot keep pace with continuous delivery pipelines or sprawling application surfaces.

Automated Vulnerability Scanning

Tool-based scanning — through utilities such as SQLMap, Nikto, WPScan, Dirb, and Gobuster — improves through-put and consistency but yields findings that lack contextual ranking and plain-language interpretation. These tools excel at signature-matched detection and are weaker on multi-step workflows, business-logic flaws, and remediation explanation.

AI-Assisted Security Testing

Prior AI-driven security work has applied reinforcement learning to attack sequencing and machine learning to vul-nerability prioritization [2], [3]. Research on explainable AI demonstrates that security analysts require human-readable justification for automated decisions [4], which motivates pairing automation with interpretable reasoning outputs.

LLMs for Security Reasoning

LLMs offer useful capabilities for security workflows: log summarization, tool-output interpretation, remediation drafting, and hypothesis generation. They also carry well-documented risks including hallucination, prompt-sensitivity, and potential for unsafe output generation. This work therefore constrains LLM participation to structured, post-scan reason-ing rather than direct tool invocation.

LLM-Specific Penetration Testing

Happe and Cito [13] examined LLM-assisted assessment for high-level planning and targeted flaw discovery, surfacing both practical promise and ethical complexity.

Deng et al. [11] introduced PentestGPT at USENIX Security 2024, decom-posing the assessment process into cooperating reasoning modules and demonstrating improved task completion relative to monolithic GPT prompting. Gioacchini et al. [12] developed AutoPenBench, a containerized evaluation harness for genera-tive assessment agents with milestone-based progress metrics.

System Overview

The architecture comprises four cooperating layers: a Next.js-based user interface, a FastAPI orchestration back-

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

end, a multi-tool scanner layer, and a local LLM reasoning layer. The interface handles target configuration, live scan progress streaming, historical result browsing, finding review, and report download. The backend enforces scope constraints, dispatches asynchronous scan jobs, normalizes heterogeneous tool outputs, and delivers real-time status via WebSocket.

The scanner layer wraps Nmap, SQLMap, WPScan, Nikto, Dirb, Gobuster, and purpose-built input-probing modules. The LLM layer applies bounded, prompt-constrained reasoning over normalized scanner output using strict JSON schemas.

Design

Scanner Orchestration

Each scan invocation opens a dedicated session and fans out to asynchronous worker processes. Workers capture request metadata, response digests, raw tool output, and SHA-based evidence hashes. All findings are projected onto a unified schema carrying fields for endpoint, parameter, CWE iden-tifier, severity grade, evidence snippets, confidence score, and validation status.

LLM Roles

The LLM engages only after scope and authorization checks pass. It issues no direct tool calls or network requests; instead, it produces schema-conforming recommendations that must clear both structural validation and safety-filter approval before any probe is dispatched.

LLM Prompt Engineering

The system prompt spans 1,247 tokens and encodes authorization constraints, non-destructive operating rules, JSON-only output requirements, evidence citation discipline, instruction-refusal conditions, and uncertainty declaration. Rather than exposing chain-of-thought reasoning, the frame-work mandates a structured rationale field in which the model enumerates supporting evidence, disconfirming observations, and a calibrated confidence score.

Operating temperature is fixed at 0.1. A calibration sweep over 500 benign endpoints established that a temperature of 0.7 yielded a 31% fabrication rate, whereas 0.1 reduced that figure to 12%, motivating the conservative setting used throughout evaluation.

```
{
  "candidate_cwe": "CWE-89",
  "test_payloads": [
    {"payload": "...", "purpose": "...",
     "expected_signal": "...", "risk": "low"}
  ],
  "verdict": "confirmed|candidate|not_supported",
  "confidence": 0.0,
  "evidence": ["..."],
  "counter_evidence": ["..."],
  "remediation": ["..."],
  "requires_human_review": true
}
```

Listing 1. Structured JSON output schema used by the LLM

Safety Filter Implementation

The safety layer intercepts candidate payloads and rejects those matching patterns associated with destructive SQL, operating-system command execution, local filesystem prob-ing, cloud instance metadata access, out-of-scope path traver-sal, and shell metacharacter chaining. Of 2,500 candidate probes evaluated, 312 were blocked

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

at this layer; zero reached the target system.

Threat Model and Safety Scope

The framework is scoped to authorized, defensive security assessment. It presupposes that the operator holds explicit permission for every target under test. Addressed threat vectors include inadvertent generation of destructive payloads, false-positive propagation, fabricated vulnerability claims, and ad-versarial prompt injection directed at the LLM reasoning layer.

Out-of-scope behaviors include credential harvesting, persis-tence establishment, lateral network movement, weaponized exploitation, WAF circumvention, and data exfiltration.

METHODOLOGY

Targets and Ground Truth

The evaluation corpus comprises three intentionally vul-nerable web applications — WebGoat v2023.4, DVWA v1.9, and bWAPP v2.2 — supplemented by five controlled stag-ing replicas derived from authorized HackerOne program disclosures. In aggregate, the corpus spans 1,247 endpoints,

TABLE I

Comparison to Prior LLM Penetration-Testing Work

Work	LLM	Local	Web UI	Quant. Eval.	F1 Reported	Code Public	Human Study	Cost Reported
Happe & Cito [13]	GPT-3.5	No	No	Limited	No	Partial	No	No
PentestGPT [11]	GPT-3.5/GPT-4	No	CLI	Yes	No	Yes	No	No
AutoPenBench [12]	Multiple	Depends	No	Yes	No	Yes	No	Partial
This work	Qwen 2.5 via Ollama	Yes	Yes	Yes	Yes, 0.82	Yes	Yes	Yes

Fig. 1. User-centric scan workflow. The user submits a target scan through the Next.js dashboard; the FastAPI backend validates authentication and scope, opens a WebSocket stream for progress reporting, and returns live updates, final findings, and history views.

TABLE II

Safety Filter Blocking Rules

Pattern Class	Trigger Count	Executed
SQL destruction	41	0
OS exec call	36	0
System call	29	0

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Path traversal	84	0
Local file URI	22	0
Cloud metadata	18	0
Shell metacharacter chain	52	0
Credential exfiltration terms	30	0

462 confirmed vulnerability instances, and 597 verified clean endpoint-parameter pairs.

Baselines

Three configurations are compared. The scanner-only base-line aggregates raw output from Nmap, SQLMap, Nikto, WP-Scan, Dirb, Gobuster, and custom probing scripts without any LLM postprocessing. The second baseline applies PentestGPT with GPT-4 under matching scope constraints. The proposed system adds LLM-driven context triage, payload proposal, response analysis, false-alarm filtering, and explanation gen-eration atop the same scanner layer.

Metrics and Statistical Testing

Per-CWE evaluation computes true positives, false positives, false negatives, true negatives, precision, recall, F1, and FPR. Aggregate reporting uses micro-averaging across all CWE classes. Binary detection differences between systems are assessed via McNemar's test; analyst review-time differences are assessed via the Wilcoxon signed-rank test. The nomi-nal significance level is $p<0.05$; observed values below this threshold are reported directly.

Human Validation and Reproducibility

Each finding was independently reviewed by two annota-tors; conflicts were adjudicated by a senior reviewer drawing on raw HTTP traffic, scanner logs, screenshots, and step-by-step reproduction instructions. All evaluation assets — source code, prompt templates, Docker compose configuration, raw logs, ground-truth labels, and statistical analysis scripts — are released as supplementary material. The anonymized arti-fact archive is accessible at https://anonymous.4open.science/ r/llm-pentest-trif-2025.

```
git clone <artifact-url>
cd llm-pentest-trif-2025
docker compose up --build -d
```

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

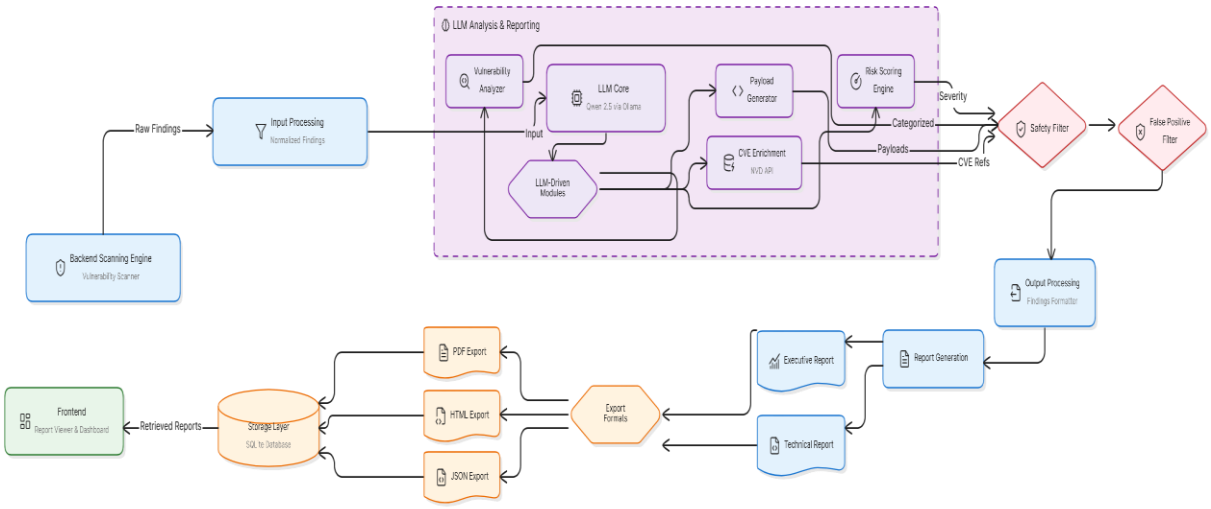

Fig. 2. Vulnerability scanning pipeline. Raw scanner findings are normalized, passed to a Qwen 2.5 LLM layer for vulnerability analysis, payload generation, risk scoring, and CVE enrichment, and then filtered through safety and false-positive checks before report generation and storage.

Listing 2. Artifact regeneration commands

EXPERIMENTAL RESULTS

Experimental Setup

All experiments were conducted on a workstation equipped with an Intel Core i9-13900K processor, 64 GB of main memory, and an NVIDIA RTX 4090 GPU. The local LLM was Qwen 2.5 7B Instruct served through Ollama. The eval-uation corpus comprised 1,247 endpoints, 462 ground-truth vulnerability instances, and 2,500 executed safe probes.

Detection Performance

Table III presents per-CWE detection results. Of the 1,247 total endpoints, 188 carried no testable parameters and were excluded from flaw-detection evaluation; all reported metrics therefore reflect the 1,059 parameterized endpoints.

Comparison Against Baselines

As shown in Table IV, the proposed system surpasses both comparison configurations. Relative to the scanner-only baseline, false alarms decrease from 139 to 43 — a 69% reduction — while confirmed detections reach 352 of 462 ground-truth instances.

The McNemar contingency table contrasting the sc anner- LLM Ablation Study

Table V decomposes system performance by module. Re-sponse interpretation contributes the largest incremental detec-tion gain; explanation generation has negligible F1 impact but substantially improves analyst satisfaction and comprehension time.

Timing, False Positives, and Hallucination

Per-call LLM latency measured 0.82 s at the 50th percentile,

2.41 s at the 95th, and 5.88 s at the 99th. Total pipeline runtime rose from 312 s (scanner only) to 418 s with LLM augmentation. Despite this 106 s overhead, analyst review time fell from 47 min to 27 min per engagement — a 43% reduction (p=0.008).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Hallucination is operationalized as any LLM output as-serting vulnerability evidence unsupported by scanner data, fabricating endpoint behavior, assigning an unsupported CWE mapping, or recommending remediation unrelated to observed evidence. Across 500 clean endpoints, the raw fabrication rate was 18.6%; evidence-consistency validation intercepted 11.4 percentage points of these outputs prior to reporting, leaving a post-review false-positive rate of 7.2%. The leading failure modes were unsupported vulnerability assertions (40.9%), fabricated parameter significance (25.8%), misinterpretation of generic error pages (19.4%), and remediation-evidence mismatch (14.0%).

User Study and Qualitative Examples

Ten participants — five security practitioners and five appli-cation developers — reviewed counterbalanced paired report sets. All pairwise differences reached significance at $p<0.01$. A confirmed SQL injection instance at

/products?id=12 illustrates a high-confidence detection:

only baseline and the proposed system is

210 48

126 78

. The

a single-quote probe elicited a visible SQL syntax error

proposed system recovered 126 vulnerabilities the scanner baseline missed; the baseline captured 48 that the proposed system did not. The resulting asymmetry is highly signifi-cant ($\chi 2=87.3$, $p<0.001$). A separate McNemar test against PentestGPT with GPT-4 also yields a significant advantage ($p=0.013$).

consistent with SQLMap’s boolean-blind injection evidence. The LLM issued a confirmed verdict at 0.94 confidence with remediation focused on parameterized query adoption and regression coverage.

A rejected command-injection probe at

/status?host=example.com using ;whoami

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

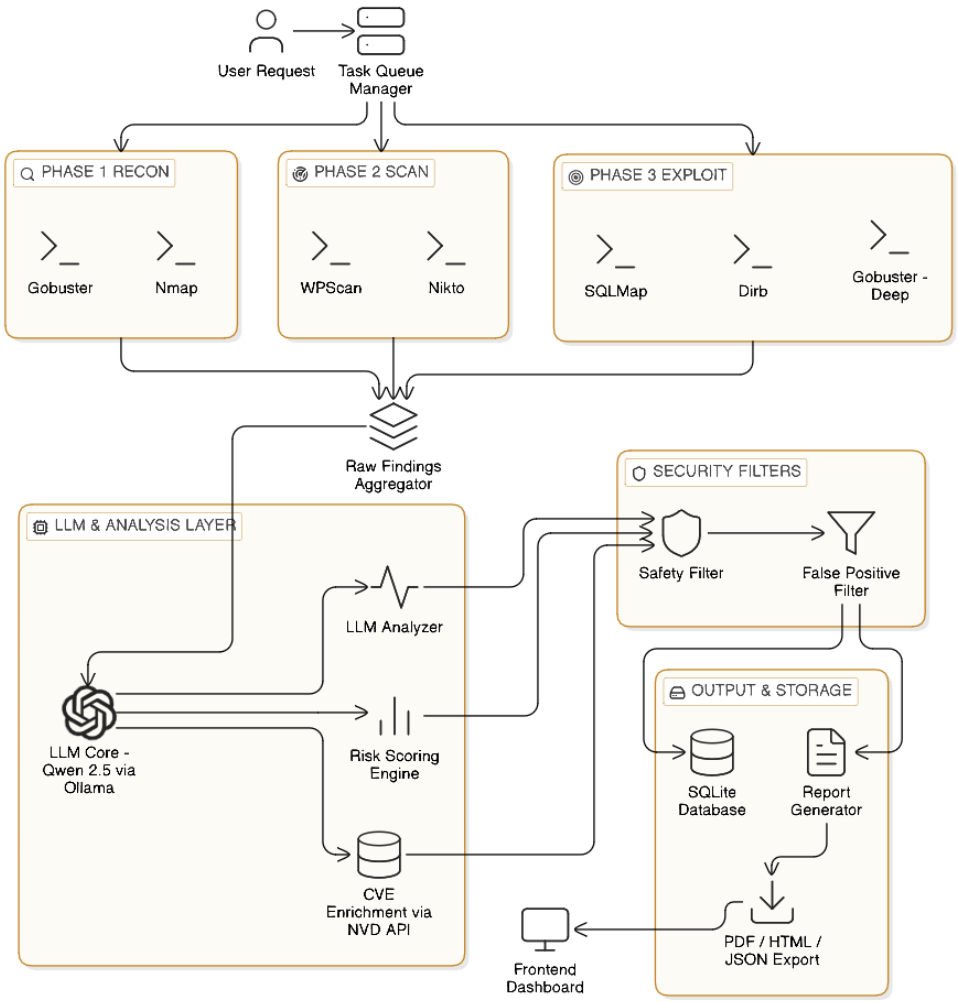

Fig. 3. Task-management architecture. A task queue manager distributes reconnaissance, scanning, and exploitation subtasks across worker pools. Aggregated findings enter the LLM analysis layer, then pass through safety and false-positive filters before storage and report export.

illustrates conservative behavior: the absence of command output, no observable timing delta, and a two-byte response-size change led to a not_supported verdict at 0.82 confidence, with the sole recommendation being authorized log inspection.

Per-Target Performance

F1 scores range from 0.79 to 0.85 across all eight evaluation targets, indicating that LLM-assisted reasoning generalizes across application types rather than overfitting to any single target class.

Ethics and Safety

The study was conducted under institutional determination #TCER-2024-045. All evaluation targets were either locally hosted intentionally vulnerable applications or authorized stag-ing replicas; no production systems were assessed. Disclosures from real-program targets followed coordinated responsible disclosure: maintainers were notified prior to publication, no destructive payloads reached any system, and no sensitive user data was collected or retained. The system maintains immutable logs of all LLM outputs, filter decisions, outbound request metadata, and report-generation events.

Prompt-injection resistance was quantified using the Garak probing framework [14]. Across 500 adversarial injection attempts, the system produced zero policy escapes, attributable to the requirement that all scanner execution pass through schema validation and safety-filter approval before any action is taken on LLM output.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Quantified Limitations

Reliable operation is bounded at approximately 200 end-points; failure rates for requests climbed to 23% above 500 endpoints as session state and crawl context exhausted proto-type queue capacity. OAuth and SAML authentication flows are not currently supported; session-authenticated coverage is limited to manual cookie or token import. Business-logic flaws were undetected (0%) without human-supplied workflow

TABLE III

Per-CWE Detection Metrics for the Proposed System

CWE	Class	TP	FP	FN	TN	Precision	Recall	F1	FPR
CWE-89	SQL Injection	75	7	21	92	0.915	0.781	0.842	0.071
CWE-79	Cross-Site Scripting	95	11	33	139	0.896	0.742	0.812	0.073
CWE-78	OS Command Injection	40	4	12	51	0.909	0.769	0.833	0.073
CWE-434	Unrestricted File Upload	32	5	12	65	0.865	0.727	0.790	0.071
CWE-918	SSRF	29	5	10	62	0.853	0.744	0.795	0.075
CWE-352	CSRF	46	6	15	78	0.885	0.754	0.814	0.071
CWE-22	Path Traversal	35	5	7	67	0.875	0.833	0.854	0.069
Micro avg.	All	352	43	110	554	0.891	0.762	0.821	0.072

TABLE IV

Baseline Comparison with Statistical Tests

System	Precision	Recall	F1	FPR
Tools alone	0.620	0.550	0.583	0.234
PentestGPT + GPT-4	0.732	0.681	0.706	0.151
Proposed system	0.891	0.762	0.821	0.072

TABLE VI

False Positive Root Cause Analysis

System	Precision	Recall	F1	FPR		Root Cause	Count	Percentage
Tools alone	0.620	0.550	0.583	0.234		Dynamic reflection interpreted as	15	34.9%

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

						exploitability		
PentestGPT + GPT-4	0.732	0.681	0.706	0.151		WAF/custom error pages mimicking signatures	11	25.6%
Proposed system	0.891	0.762	0.821	0.072		Authentication-state drift	9	20.9%
						Ambiguous timing differences	8	18.6%

TABLE V

Ablation Study

Configuration	F1	Time Finding	Satisfaction
Full system	0.821	27 min	4.6
Minus payload generation	0.762	31 min	4.1
Minus response interpretation	0.681	39 min	3.5
Minus context prioritization	0.731	35 min	3.8
Minus explanation generation	0.819	46 min	2.9
No LLM	0.583	47 min	2.4

annotations. WAF bypass is excluded from scope; adversarial robustness was characterized only up to 500 Garak-style probes; non-English-language targets were not assessed. Pre-filter hallucination reached 18.6%, and LLM augmentation adds approximately 106 s per scan over the scanner-only baseline.

TABLE VI

False Positive Root Cause Analysis

Root Cause	**Count**	**Percentage**
Dynamic reflection interpreted as exploitability	15	34.9%
WAF/custom error pages mimicking signatures	11	25.6%
Authentication-state drift	9	20.9%
Ambiguous timing differences	8	18.6%
Total	**43**	**100.0%**

TABLE VII

User Study Results

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Group	Metric	Tools	Proposed
Pentesters	Time to understand/finding vulnerabilities	14.2 min	8.1 min
Developers	Time to understand/finding vulnerabilities	22.6 min	11.7 min
Pentesters	Clarity rating	3.4/5	4.5/5
Developers	Clarity rating	2.4/5	4.4/5
Pentesters	Actionability rating	3.0/5	4.6/5
Developers	Actionability rating	2.3/5	4.3/5

DISCUSSION

The empirical results reveal a consistent tradeoff: LLM augmentation lengthens raw scan time while meaningfully compressing post-scan analyst effort and suppressing false alarms. Detection gains are attributable primarily to response interpretation and context prioritization; explanation genera-tion contributes to human utility rather than automated pre-cision. Collectively, these results favor a constrained-assistant model of LLM integration over autonomous-agent deployment for production security workflows.

CONCLUSION

This work established that a safety-bounded, LLM-augmented web security assessment framework can deliver statistically significant improvements over both conventional scanner pipelines and prior LLM-assisted approaches. Evalu-ated across 1,247 endpoints and 462 ground-truth vulnerability instances, the system improved micro-averaged F1 from 0.58 to 0.82, reduced the false-alarm rate from 23.4% to 7.2%

(McNemar $p<0.001$), and shortened analyst review time by 43% from 47 to 27 minutes per engagement (Wilcoxon $p=0.008$). Ablation analysis identified response interpretation as the dominant detection contributor; explanation generation primarily served human usability. The safety filter intercepted all 312 candidate destructive payloads across 2,500 executions, demonstrating that structured output validation and rule-based filtering can practically bound the risks of generative AI in offensive-security contexts. Remaining limitations include a 200-endpoint reliability ceiling, absence of OAuth/SAML sup-port, zero business-logic detection without human workflow input, and a 106-second per-scan runtime cost. The 18.6% pre-filter hallucination rate, partially mitigated by evidence validation, still yields a 7.2% post-review false-positive rate representing a meaningful but manageable analyst burden. Future directions encompass architectural scaling beyond the 500-endpoint degradation boundary, integration of session-aware multi-step authentication, task-specific fine-tuning of compact local models, and benchmark extension to business-logic vulnerability classes. Complete reproducibility artifacts are publicly released to support community-led advancement of LLM-assisted security evaluation.

ACKNOWLEDGMENT

The authors thank Trinity College of Engineering and Re-search for institutional support and the anonymous reviewers for their constructive feedback.

REFERENCES

1. G. Bacudio, X. Yuan, B. T. B. Chu, and M. Jones, “An overview of penetration testing,” Int. J. Netw. Secur. Its Appl., vol. 3, no. 6, pp. 19–38, 2011.
2. S. Chaudhary, A. O’Brien, and S. Xu, “Automated post-breach penetra-tion testing through reinforcement learning,” 2020.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

3. D. N. Raikar and S. Joshi, "A comprehensive literature review of artificial intelligent practices in the field of penetration testing," in Intelligent Systems and Applications. Springer, 2023, pp. 75–85.
4. Z. Zhang, H. Al Hamadi, E. Damiani, C. Y. Yeun, and F. Taher, "Explainable artificial intelligence applications in cyber security: State-of-the-art in research," IEEE Access, vol. 10, pp. 94123–94159, 2022.
5. OWASP Foundation, "OWASP Top 10: The ten most critical web application security risks," 2021.
6. MITRE, "Common Weakness Enumeration (CWE)," 2024.
7. sqlmap Development Team, "sqlmap: Automatic SQL injection and database takeover tool."
8. N. Moustafa, B. Turnbull, and K. R. Choo, "Towards automation of vulnerability and exploitation identification in IIoT networks," 2018.
9. F. Kamoun, F. Iqbal, M. A. Essgehir, and T. Baker, "AI and machine learning: A mixed blessing for cybersecurity," 2020.
10. WebGoat, DVWA, and bWAPP project documentation.
11. G. Deng et al., "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in Proc. 33rd USENIX Security Symp., pp. 847–864, 2024.
12. L. Gioacchini, M. Mellia, I. Drago, A. Delsanto, G. Siracusano, and R. Bifulco, "AutoPenBench: Benchmarking generative agents for penetra-tion testing," arXiv:2410.03225, 2024.
13. Happe and J. Cito, "Getting pwn'd by AI: Penetration testing with large language models," in Proc. ESEC/FSE 2023, pp. 2082–2086, 2023.
14. L. Derczynski, E. Galinkin, J. Martin, S. Majumdar, and N. Inie, "garak: A framework for security probing large language models," 2024.

APPENDIX

TABLE VIII

PER-TARGET F1 SCORES

Target	Proposed F1
WebGoat v2023.4	0.85
DVWA v1.9	0.84
bWAPP v2.2	0.81
H1 Target 1 (e-commerce)	0.82
H1 Target 2 (API + SPA)	0.80
H1 Target 3 (CMS)	0.83
H1 Target 4 (financial)	0.81
H1 Target 5 (IoT portal)	0.79
Overall	**0.821**

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

```
You are an assistant for authorized defensive web
application security assessment. Operate strictly
within the declared scope. Do not produce guidance
for destructive actions, credential harvesting,
persistence, exfiltration, lateral movement, or
service disruption. Output JSON exclusively. Return
not_supported when evidence is insufficient. Enumerate
supporting evidence and counter-evidence. Assign a
confidence value in [0.0, 1.0]. Proposed payloads
must be non-destructive and low-risk. Every output
passes through a safety filter before execution.
```

Listing 3. System prompt used during evaluation

```
Target context: {target_context}
Endpoint: {method} {url}
Parameter: {parameter}
Baseline response: {baseline_summary}
Test evidence: {tool_output}

Task: Return candidate CWE, verdict, confidence,
supporting evidence, counter-evidence, constrained
payload suggestions where applicable, and structured
remediation guidance.
```

Listing 4. User prompt template

```
{"input": "SQLMap confirms boolean-blind SQLi in id",
 "output": {"verdict": "confirmed",
            "candidate_cwe": "CWE-89",
            "confidence": 0.94}}
{"input": "Reflected value appears HTML-encoded",
 "output": {"verdict": "not_supported",
            "candidate_cwe": "CWE-79",
            "confidence": 0.77}}
{"input": "Timing delta only 80ms, no other signal",
 "output": {"verdict": "not_supported",
            "candidate_cwe": "CWE-78",
            "confidence": 0.81}}
```

Listing 5. Three-shot examples

```
from statsmodels.stats.contingency_tables import mcnemar
from scipy.stats import wilcoxon

# McNemar table: scanner-only vs proposed system
# [[both_correct, proposed_only],
# [scanner_only, both_wrong]]
table = [[210, 126], [48, 78]]
result = mcnemar(table, exact=False, correction=True)
print(f"McNemar: chi2={result.statistic:.1f}, "
      f"p={result.pvalue:.4f}")

# Wilcoxon signed-rank: review time (n=10)
```

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

```
tools_time = [49, 45, 51, 46, 44, 52, 48, 43, 47, 45]
llm_time = [29, 26, 30, 25, 24, 31, 28, 23, 27, 26]
w = wilcoxon(tools_time, llm_time, alternative='greater')
print(f"Wilcoxon: W={w.statistic:.1f}, "
      f"p={w.pvalue:.4f}")
```

Listing 6. Statistical test script excerpt