APEX: Agentic Pentesting Execution

Tuigg R. Barcelos[1], Camilla B. Quincozes[2], Gabriel Pereira Bellagamba[1], Paulo Souza[1], Silvio E. Quincozes[1,2]

[1]AI Horizon Labs, PPGES, Federal University of Pampa (UNIPAMPA)
[2]PPGCO, Federal University of Uberlandia (UFU)
{tuiggbarcelos, camillaborchhardt, gabrielbellagamba}.aluno@unipampa.edu.br

{silvioquincozes, paulosilas}@unipampa.edu.br

Abstract. *Penetration testing validates whether software systems resist realistic attacks, but it remains costly, expertise-intensive, and difficult to scale. Recent advances in Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), and tool-using agents enable pentesting to be reframed as a controlled agentic workflow. This paper proposes APEX, an Agentic Pentesting Execution architecture that combines policy-bounded planning, specialized agents, constrained tool execution, evidence tracking, human approval, risk triage, and remediation-oriented reporting. APEX aims to support authorized assessments with greater traceability, reproducibility, safety, and educational value.*

1. Introduction

Software systems increasingly expose web interfaces, Application Programming Interfaces (APIs), cloud services, mobile components, dependency chains, and Internet of Things (IoT) devices to hostile environments. In this setting, secure software engineering must combine preventive practices, such as secure coding, code review, and threat modeling, with adversarial validation mechanisms that reveal how weaknesses can be exploited in practice. Penetration testing, commonly referred to as pentest, addresses this need by simulating realistic attacks under authorized and controlled conditions. A pentest typically includes scoping, reconnaissance, vulnerability analysis, exploitation, evidence collection, reporting, and remediation guidance, helping teams identify vulnerabilities, estimate impact, validate defenses, and prioritize fixes before real attackers exploit them [Garg and Bansal 2021, Adam et al. 2023, OWASP Foundation 2024, Scarfone et al. 2008].

AI threats reinforce the motivation. Agents can select targets and synthesize attacks [Guan et al. 2026], widening the gap between automated attackers and expert-dependent defenders. This motivates controlled, auditable assessment. Recent work on Large Language Models (LLMs) and tool-using agents indicates that this transition is technically plausible, but still immature. PentestGPT shows that decomposing pentesting into specialized modules improves task completion compared with direct LLM usage [Deng et al. 2024]. PentestAgent explores multi-agent collaboration and Retrieval-Augmented Generation (RAG) to improve domain knowledge during automated assessments [Shen et al. 2025]. APT-Agent emphasizes memory and hallucination correction for long-running exploitation workflows [Li et al. 2026]. However, PentestEval reports that end-to-end autonomous pentesting remains fragile, with important limitations across decomposed stages of the workflow [Yang et al. 2025]. These results suggest that the

central challenge is not simply automating attacks, but designing controlled agentic workflows that combine planning, retrieval, constrained tool execution, evidence tracking, remediation feedback, and human approval.

This paper proposes APEX, an Agentic Pentesting Execution architecture for secure software engineering. APEX frames agentic pentesting as an authorized assessment workflow in which specialized LLM-based agents plan, execute, interpret, triage, and report security tests under explicit scope, policy, evidence, and approval constraints. The contribution is threefold: (i) we reformulate classical pentesting as an auditable agentic process; (ii) we propose a governed integration of agentic planning, tool mediation, evidence, and human oversight that makes authorization, traceability, safety, and remediation central design elements; and (iii) we define expected results and a future work plan that prioritizes vulnerable web applications and course-oriented projects, with cyber-range assessment as a later stage.

2. Related Work

Traditional penetration testing literature focuses on mature processes, clear scope, tool support, and reproducible reporting. Garg and Bansal [Garg and Bansal 2021] review pentesting as a risk mitigation practice, while Adam et al. [Adam et al. 2023] summarize frameworks, tools, and application areas. Standards and guides such as OWASP WSTG, PTES, OSSTMM, NIST SP 800-115, and ISO/IEC 15408 provide structured assessment guidance [OWASP Foundation 2024, Penetration Testing Execution Standard 2014, Herzog 2010, Scarfone et al. 2008, International Organization for Standardization 2022, Spínola 2004]. However, these references remain human-centered and do not define how agentic systems should coordinate tools, memory, evidence, remediation, and approval gates.

LLM-enabled penetration testing has recently received growing attention. PentestGPT introduces a modular design to reduce context loss during multi-step assessments [Deng et al. 2024], while PentestAgent uses multi-agent collaboration and RAG to improve domain knowledge and task automation [Shen et al. 2025]. HackSynth proposes a planner-and-summarizer architecture for Capture-the-Flag-style benchmarks [Muzsai et al. 2024], and PenHeal connects automated pentesting with remediation recommendation [Huang and Zhu 2024]. APT-Agent and planning-based approaches further indicate that memory, hallucination reduction, and structured planning are central to long-running exploitation workflows [Li et al. 2026, Wang et al. 2025].

Research on language agents provides additional foundations for this direction. ReAct, Reflexion, Toolformer, and AutoGen show how reasoning, memory, tool use, and multi-agent coordination can support complex tasks [Yao et al. 2023, Shinn et al. 2023, Schick et al. 2023, Wu et al. 2023]. However, agentic cybersecurity systems also introduce risks such as hallucinated commands, prompt injection, unsafe tool use, and loss of control over long action chains [Greshake et al. 2023, OWASP Foundation 2025].

The literature therefore reveals three gaps that matter for secure software engineering. First, many systems prioritize task completion without making authorization, evidence custody, and human approval central architectural concerns. Second, remediation is often treated as a final text output rather than as a feedback loop connected to development artifacts and retesting. Third, evaluation frequently focuses on vulnerable machines or

challenge environments, which are insufficient to characterize safety, reproducibility, and usefulness in development workflows. APEX addresses these gaps through policy and approval controls, an Evidence Store linking actions to findings, remediation-driven retesting, and staged evaluation of safety, reproducibility, and usefulness.

3. APEX: Agentic Pentesting Execution

APEX is designed for authorized security assessment in software engineering contexts. Its goal is to assist testers, students, and development teams in conducting structured pentests without replacing human responsibility. The architecture follows four design principles: explicit authorization, controlled tool mediation, evidence-driven reasoning, and remediation-oriented reporting. As shown in Figure 1, APEX is organized into four connected layers: governance and control, agentic planning and tool mediation, bounded testing actions, and evidence, risk, and remediation.

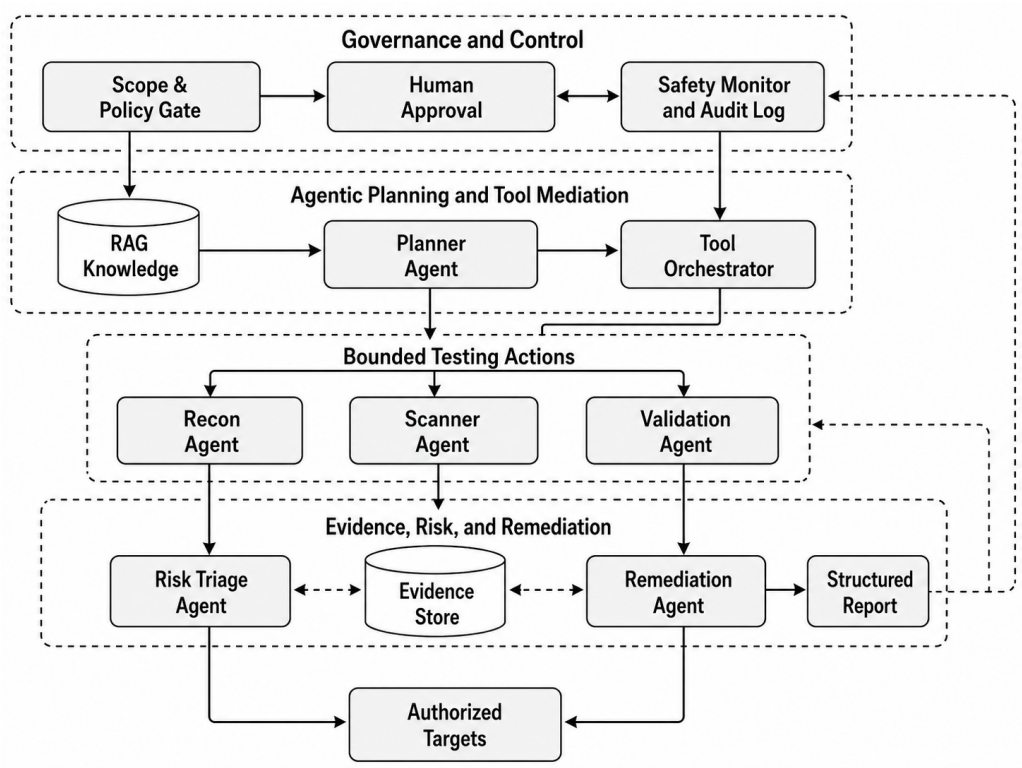

Figure 1. APEX architecture for agentic pentesting.

The *Governance and Control* layer defines the boundaries of the assessment. The *Scope and Policy Gate* represents the rules of engagement, including authorized targets, allowed credentials, test windows, forbidden actions, rate limits, approval requirements, and tool permissions. The *Human Approval* component introduces explicit oversight for sensitive decisions, while the *Safety Monitor and Audit Log* supervises execution and records relevant decisions. Together, these components prevent the architecture from acting as an unconstrained attacker and make authorization, accountability, and auditability first-class concerns.

The *Agentic Planning and Tool Mediation* layer transforms the authorized assessment scope into executable tasks. The *RAG Knowledge* component provides retrieval-

augmented access to security references, vulnerability taxonomies, tool documentation, previous reports, and project-specific requirements, including OWASP, PTES, MITRE ATT&CK, CWE, CVE, and CVSS [Lewis et al. 2020, MITRE Corporation 2024c, MITRE Corporation 2024b, MITRE Corporation 2024a, FIRST 2023]. The *Planner Agent* uses this knowledge to decompose the assessment into activities such as reconnaissance, scanning, validation, evidence collection, and reporting. The *Tool Orchestrator* checks each proposed call against authorized targets, tool permissions, forbidden actions, rate limits, and approval requirements. It then blocks the call, requests human approval, or forwards it for execution. This separation between planning and execution is inspired by agentic patterns such as ReAct, Reflexion, Toolformer, and multi-agent orchestration [Yao et al. 2023, Shinn et al. 2023, Schick et al. 2023, Wu et al. 2023].

The *Bounded Testing Actions* layer groups the specialized agents responsible for operational testing. The *Recon Agent* collects passive and active information within the authorized scope. The *Scanner Agent* invokes approved tools for network, web, dependency, and API analysis. The *Validation Agent* checks whether suspected findings are reproducible and exploitable under safe constraints. All actions remain mediated by the orchestrator, constrained by policy, and limited to authorized targets, preserving operational value without enabling unrestricted offensive automation.

The *Evidence, Risk, and Remediation* layer converts raw observations into actionable security outcomes. The *Evidence Store* preserves structured artifacts such as commands, outputs, target metadata, suspected weaknesses, confidence levels, and validation records. The *Risk Triage Agent* correlates this evidence with vulnerability taxonomies and severity models, supporting prioritization and reducing false positives. The *Remediation Agent* translates validated findings into developer-oriented actions, including affected components, suggested fixes, retesting steps, and references to standards or project requirements. The workflow culminates in a *Structured Report*, which consolidates all output generated from the entire process, including executive summaries, technical evidence, risk prioritization, and remediation guidance.

The dashed feedback paths in Figure 1 represent control and learning loops. Evidence and reports can trigger additional review, update the planning context, or require human approval before higher-risk actions are performed. These loops are important because penetration testing is rarely linear: observations from one stage often change the next testing decision, require scope verification, or demand additional validation. APEX therefore models pentesting as a governed iterative workflow, combining agentic planning with explicit safety constraints, evidence custody, and remediation feedback.

APEX can operate in three modes. In learning mode, it explains each step and prioritizes safe demonstrations in controlled laboratories. In assisted-professional mode, it accelerates reconnaissance, triage, evidence organization, and report drafting while leaving high-impact actions to the tester. In continuous-assessment mode, it periodically validates known controls in development or staging environments under a restricted policy. These modes allow the same architecture to support education, software engineering, and operational security without assuming full autonomy.

4. Conclusion and Future Work Plan

This paper proposed APEX, an Agentic Pentesting Execution architecture for secure software engineering. APEX reformulates penetration testing as a governed agentic workflow in which specialized LLM-based agents support planning, tool execution, evidence organization, risk triage, and remediation-oriented reporting under explicit authorization, policy, and human-approval constraints. The goal is not unrestricted offensive autonomy, but safer and more reproducible support for authorized assessments conducted by testers, developers, and educators.

The expected results are grounded in recent evidence from LLM-based penetration-testing systems. PentestGPT shows that decomposing pentesting into specialized modules improves task completion compared with direct LLM usage [Deng et al. 2024]; PentestAgent suggests that multi-agent collaboration and Retrieval-Augmented Generation (RAG) can improve domain knowledge and task automation [Shen et al. 2025]; and PenHeal indicates that connecting pentesting with remediation can improve vulnerability coverage and repair effectiveness in controlled experiments [Huang and Zhu 2024]. These findings support the design hypothesis behind APEX: agentic pentesting should combine planning, retrieval, memory, constrained tool execution, evidence tracking, human approval, and remediation feedback instead of relying on a monolithic autonomous agent.

APEX is expected to improve traceability, reproducibility, safety, and learning value. Traceability comes from linking findings to evidence, tool outputs, policy decisions, and remediation suggestions. Reproducibility comes from storing task state, parameters, observations, and execution artifacts in a structured format. Safety comes from authorization policies and approval gates for actions that may affect availability, expose credentials, modify data, or increase operational risk. Learning value comes from explaining each step, the risk it evaluates, and its relation to secure development practices.

The future work plan will implement and evaluate APEX in four stages: (i) implementing the policy layer, orchestrator, evidence store, and basic agents for reconnaissance, scanning, and reporting; (ii) integrating retrieval-augmented knowledge from OWASP, CWE, CVE, MITRE ATT&CK, CVSS, tool manuals, and project-specific requirements; (iii) adding validation and remediation agents with approval gates for higher-risk actions; and (iv) initially evaluating the architecture in controlled web applications and course-oriented projects, with cyber ranges and continuous assessment reserved for later work.

The evaluation will investigate whether policy-bounded agents increase vulnerability discovery coverage without increasing unsafe actions; whether structured evidence improves report quality and reduces false positives; whether remediation-oriented agents help developers understand, prioritize, and fix vulnerabilities; and whether agentic pentesting improves learning outcomes when compared with conventional tool-based laboratory activities. Metrics will include vulnerability coverage, confirmed exploitability, false positive rate, time to report, evidence completeness, remediation usefulness, human interventions, policy violations, and perceived learning value. All evaluations will follow explicit scope, logging, and safety constraints.

References

Adam, H. M., Widyawan, and Putra, G. D. (2023). A review of penetration testing frameworks, tools, and application areas. In *2023 IEEE 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pages 319–324. IEEE.

Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., and Rass, S. (2024). PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *Proceedings of the 33rd USENIX Security Symposium*. USENIX Association.

FIRST (2023). Common Vulnerability Scoring System Version 4.0. `https://www.first.org/cvss/v4.0/`. Accessed: 2026-06-05.

Garg, D. and Bansal, N. (2021). A systematic review on penetration testing. In *2021 2nd Global Conference for Advancement in Technology (GCAT)*. IEEE.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., and Fritz, M. (2023). Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*. ACM.

Guan, J., Blanchard, T., Foerster, H., Jia, H., Huang, G., and Papernot, N. (2026). AI Agents Enable Adaptive Computer Worms.

Herzog, P. (2010). *OSSTMM 3: The Open Source Security Testing Methodology Manual*. ISECOM.

Huang, J. and Zhu, Q. (2024). PenHeal: A two-stage llm framework for automated pentesting and optimal remediation. In *Proceedings of the 2nd ACM Workshop on Secure and Trustworthy Large Language Models*. ACM.

International Organization for Standardization (2022). ISO/IEC 15408: Information security, cybersecurity and privacy protection – evaluation criteria for it security. `https://www.iso.org/standard/72891.html`. Accessed: 2026-06-05.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Li, W. G., Abuadbba, A., Moore, K., and Kim, D. D. (2026). APT-Agent: Automated penetration testing using large language models.

MITRE Corporation (2024a). Common Vulnerabilities and Exposures (CVE). `https://www.cve.org/`. Accessed: 2026-06-05.

MITRE Corporation (2024b). Common Weakness Enumeration (CWE). `https://cwe.mitre.org/`. Accessed: 2026-06-05.

MITRE Corporation (2024c). MITRE ATT&CK. `https://attack.mitre.org/`. Accessed: 2026-06-05.

Muzsai, L., Imolai, D., and Lukács, A. (2024). HackSynth: Llm agent and evaluation framework for autonomous penetration testing.

OWASP Foundation (2024). Web Security Testing Guide. https://owasp.org/www-project-web-security-testing-guide/. Accessed: 2026-06-05.

OWASP Foundation (2025). OWASP Top 10 for Large Language Model Applications. https://owasp.org/www-project-top-10-for-large-language-model-applications/. Accessed: 2026-06-05.

Penetration Testing Execution Standard (2014). The Penetration Testing Execution Standard. https://www.pentest-standard.org/. Accessed: 2026-06-05.

Scarfone, K., Souppaya, M., Cody, A., and Orebaugh, A. (2008). Technical guide to information security testing and assessment. Technical Report Special Publication 800-115, National Institute of Standards and Technology.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36.

Shen, X., Wang, L., Li, Z., Chen, Y., Zhao, W., Sun, D., Wang, J., and Ruan, W. (2025). PentestAgent: Incorporating llm agents to automated penetration testing. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*. ACM.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36.

Spínola, R. O. (2004). Conhecendo a iso 15408: Trabalhando com segurança da informação. *Engenharia de Software Magazine*, 45:31–36.

Wang, L., Shi, X., Li, Z., Jiang, Y., Tan, S., Jiang, Y., Cheng, J., Chen, W., Shen, X., Li, Z., and Chen, Y. (2025). Automated penetration testing with llm agents and classical planning.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. (2023). AutoGen: Enabling next-gen llm applications via multi-agent conversation.

Yang, R., Cheng, M., Deng, G., Zhang, T., Wang, J., and Xie, X. (2025). PentestEval: Benchmarking llm-based penetration testing with modular and stage-level design.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*.