infiltra.ai

WHITE PAPER

The Offensive AI Manifesto

DAST vs Manual vs Agentic Pentesting — what changes, what doesn't, and how to evaluate the outputs

Prepared For:
AppSec, Engineering, and Security Leaders

Date:
March 2026

Executive Summary

Modern application security programmes generate more findings than teams can realistically act on. Detection is no longer the main challenge. The real problem is establishing trustworthy proof, prioritising what matters, and verifying fixes reliably.

KEY MESSAGE

Security validation must run at the same cadence as software delivery.

In agile environments where software changes weekly or even daily, point-in-time testing quickly becomes outdated. A vulnerability assessment performed today may no longer reflect the system after the next release. Security teams therefore need validation that operates continuously, not a process that runs once or twice a year.

This paper clarifies three testing approaches that are often conflated:

- DAST provides broad automated coverage but often produces signals that still require validation.
- Manual pentesting delivers deep contextual analysis and high-confidence findings, but it cannot scale with modern release cadence.
- Agentic pentesting uses autonomous tool-driven workflows to validate vulnerabilities, generate reproducible evidence, and create retests that verify remediation.

The central argument is simple: security validation must run at the same cadence as software delivery. Testing should produce auditable evidence, prioritise issues based on real impact, and continuously re-verify fixes so assurance does not decay as systems evolve.

KEY TAKEAWAYS

- DAST scales discovery but frequently produces signals that require additional validation.
- Manual pentesting provides deep insight but cannot keep pace with modern release cycles.
- Agentic pentesting enables continuous validation by chaining actions across workflows and producing reproducible evidence.
- Operational priority: measure security by proof and retests, not vulnerability counts.

The DevOps Reality: Why Point-in-Time Testing Plateaus

Modern applications change continuously. APIs evolve, authorisation rules shift, integrations change, and workflows move across services.

Software delivery has also accelerated. Teams now release weekly, daily, and sometimes multiple times per day. Yet security testing is still often tied to scheduled assessments.

KEY MESSAGE

Point-in-time assurance decays quickly in fast-release environments. Security validation must run with delivery.

That creates a simple problem: a pentest only reflects the system as it existed during the test window. In fast-moving environments, that assurance quickly becomes stale.

High-performing engineering teams deploy far more frequently than traditional organisations. In that model, security validation cannot remain calendar-based. It has to run continuously so findings stay relevant and fixes are verified as systems evolve.

973×

Elite performers in Google Cloud's DORA research deploy code up to 973 times more frequently than low performers, underscoring how quickly software delivery can outpace periodic assurance.

Source: Google Cloud / DORA

A pentest tied to a calendar cannot keep pace with software tied to a pipeline.

Three Modes of Engagement

Security testing today generally falls into three engagement models. Each serves a different purpose.

KEY MESSAGE

DAST provides breadth, manual pentesting provides depth, and agentic testing enables continuous validation with proof.

DAST: BROAD DYNAMIC SCANNING

DAST tests a running application by crawling endpoints and mutating inputs to detect abnormal behaviour. Its strength is scale. It can cover large application surfaces quickly and repeatedly.

However, DAST primarily produces signals. These signals often require additional validation to determine whether they represent real, exploitable vulnerabilities.

MANUAL PENTESTING: DEEP HYPOTHESIS TESTING

Manual pentesting relies on human expertise to analyse application behaviour, model trust boundaries, and test complex workflows.

This approach remains the most effective method for identifying nuanced issues such as business logic flaws or complex authorisation failures. Its limitation is cadence. Human-led testing cannot scale to match the speed of modern release cycles.

AGENTIC PENTESTING: AUTONOMOUS VALIDATION

Agentic pentesting uses autonomous agents that interact with applications through real tools such as browsers, HTTP clients, and fuzzers. These systems can chain actions across identities and workflows to validate vulnerabilities and produce reproducible evidence.

The key advantage is cadence. Validation can run continuously, generating evidence and retests that verify fixes as systems evolve.

Choose the method that matches the decision, not the method that produces the most output.

Signals vs Proof

Modern security programmes generate large volumes of findings, yet many teams still struggle to prioritise what matters. The difficulty is not detection. It is determining which findings represent real, exploitable risk.

KEY MESSAGE

Security teams do not lack alerts. They lack proof.

Many security tools produce signals—indicators that something may be wrong. Signals are useful for exploration, but they do not establish whether a vulnerability is exploitable or what its real impact might be.

Security decisions require proof. Proof demonstrates that a vulnerability can be reproduced safely and that it leads to a meaningful security outcome.

Without a shared standard for evidence, teams spend time debating findings instead of fixing them.

THE EVIDENCE LADDER

The Evidence Ladder provides a simple framework for distinguishing early indicators from validated risk.

Signal: A suspicious indicator that suggests a potential issue.

Confirmed: The behaviour can be reproduced safely with supporting evidence.

Impact: The issue demonstrates a meaningful security consequence.

Retest: An automated replay verifies that remediation has fixed the issue and prevents regression.

Operating Rule: Signals guide investigation. Confirmed issues with impact drive decisions. Retests ensure that fixes remain effective over time.

Security programmes fail when alerts are treated as truth instead of evidence.

Strategic Selection: Choosing the Right Validation Approach

Security teams often debate tools rather than outcomes. The more practical question is which approach produces the level of confidence required for the decision at hand.

KEY MESSAGE

Use the right validation method for the decision you are trying to make.

DAST: SCALABLE DISCOVERY

DAST is most useful for broad discovery across large application surfaces. It can identify potential weaknesses and exposure changes quickly, making it effective as an early signal layer. However, DAST findings often require additional validation before they can drive remediation decisions.

MANUAL PENTESTING: TARGETED DEEP VALIDATION

Manual testing is best applied where interpretation and domain understanding are required. This includes complex workflows, business logic, and high-risk systems where exploitation paths depend on application behaviour. Because it relies on human expertise, manual testing is most effective when used selectively rather than as a universal control.

AGENTIC VALIDATION: CONTINUOUS PROOF

Agentic testing provides automated validation that produces reproducible evidence and retests. This allows security verification to run repeatedly as systems change. In environments with frequent releases, this model enables validation to operate continuously rather than periodically.

Choose the method that matches the decision, not the method that produces the most output.

Economics

Comparing security tools on licence price alone is misleading. The more useful comparison is total cost of ownership: subscription cost plus the human effort required to configure the tool, maintain it, triage outputs, and re-verify fixes.

KEY MESSAGE

The value of security testing depends on whether teams can identify, prioritise, and fix the issues that actually matter.

The same applies when comparing platform-based validation with manual pentesting. A one-off manual pentest may appear cheaper in isolation, but it is not a like-for-like comparison when the objective is ongoing validation, repeated retesting, and assurance across frequent release cycles.

Outcome quality also shapes programme economics. When outputs are noisy, security teams spend time validating findings, engineering teams spend time reproducing them, and remediation effort can be diverted toward issues that are low impact or not real. At the same time, critical vulnerabilities may be delayed or overlooked.

Programmes that produce credible evidence and clearer prioritisation allow teams to focus effort where it matters and verify that fixes remain effective over time.

30×–100×

Widely cited Systems Sciences / NIST-style remediation economics show that fixing vulnerabilities late in the lifecycle can cost tens of times more than fixing them during design or development, reinforcing why timing and validation quality matter.

Source: Systems Sciences / NIST-style remediation cost research

The cost of poor security testing is not only wasted effort. It is also the risk of missing what matters.

From Reports to Continuous Validation

Traditional pentesting produces a report tied to a specific testing window. That report is valuable, but it is still a snapshot.

KEY MESSAGE

The value of offensive security shifts from producing reports to maintaining continuous validation.

Continuous validation changes the model. Instead of relying on periodic reports, teams gain a capability that can repeatedly verify vulnerabilities, confirm that fixes remain effective, and track how security posture changes over time.

This shift also reflects a practical constraint. Traditional pentesting is limited by human throughput. Every engagement requires coordination, specialised expertise, and time to execute and validate findings, which restricts how frequently testing can realistically occur as systems and release cycles accelerate.

Continuous validation also improves oversight. It enables trend analysis, clearer reporting, and better executive visibility into whether risk is reducing, recurring, or growing across releases.

This turns testing from a periodic assessment into a continuous security control.

Security testing becomes more valuable when it operates as a control, not just a report.

CI/CD Integration and Governance

In agile environments, security validation must operate alongside the delivery pipeline. Findings that represent real risk should influence release decisions in the same way as failing tests or quality checks.

Continuous validation enables this by producing reproducible evidence that can be used to enforce clear release criteria.

KEY MESSAGE

Security becomes most effective when validation results are integrated directly into release decisions.

Using evidence-driven criteria helps maintain engineering trust in security controls. Releases are not blocked by speculative alerts, but by confirmed vulnerabilities with clear impact.

This approach also clarifies governance responsibilities. Security teams define evidence standards, severity classification, and exception processes, while automated validation enforces those rules consistently.

CRITICAL	Confirmed vulnerability with demonstrated impact. Release should be blocked until remediation or an approved exception.
HIGH	Requires remediation or formally accepted risk with defined timelines.
MED / LOW	Tracked through remediation SLAs and monitored for trends.

Security controls are trusted when they block releases based on evidence, not alerts.

The Invariants of Offensive Security

Offensive security continues to evolve as software architectures, delivery models, and attack techniques change. Yet several core principles remain constant regardless of how testing is performed.

KEY MESSAGE

The methods of offensive security evolve, but its principles remain constant.

Testing must operate within clearly defined rules of engagement. Scope, rate limits, and safety controls determine what actions are permitted during testing and ensure that validation activities do not disrupt production systems.

Evidence remains the standard for establishing truth. Claims about vulnerabilities must be supported by reproducible proof demonstrating how an issue can be exploited safely and what impact it creates.

Risk is inherently contextual. The severity of a vulnerability depends not only on technical behaviour but also on how systems are used, what data they expose, and which business processes they affect.

Agentic testing does not replace these principles. Instead, it introduces new ways to execute offensive testing while preserving the same expectations of safety, evidence, and contextual risk evaluation.

The techniques of offensive security change, but the principles of safety, evidence, and contextual risk remain constant.

Agentic Testing in Practice

Agentic testing uses autonomous systems to interact with applications using the same tools and workflows a human tester would use.

These systems can navigate applications through browsers, send crafted requests to APIs, and chain actions across identities and workflows. Rather than simply scanning inputs for anomalies, they attempt to validate vulnerabilities by executing realistic attack paths.

KEY MESSAGE

Agentic testing extends offensive security by enabling continuous validation through autonomous execution of attack paths.

This approach becomes especially valuable for harder classes of security issues, such as authorisation failures, authentication and session boundary weaknesses, workflow abuse, and multi-step exploit chains that require coordinated actions across requests or user roles.

- Authorisation failures
- Session boundary issues
- Workflow abuse
- Multi-step exploit chains

Because these agents can operate continuously, they can repeatedly test applications as systems evolve and releases are deployed. This allows organisations to verify vulnerabilities, confirm that fixes remain effective, and maintain assurance as systems change.

Agentic testing does not replace human expertise. It augments it by automating repeatable validation tasks and allowing offensive testing to run far more frequently than purely human-led engagements.

API1:2023

OWASP ranks Broken Object Level Authorization as the number one risk in the 2023 API Security Top 10, directly reinforcing why object-level access control failures remain a critical class for modern validation.

Source: OWASP API Security Top 10

Agentic testing combines the reasoning of pentesting with the repeatability of automation.

How Infiltra Enables Continuous Validation

Infiltra operationalises agentic testing to deliver continuous, evidence-based security validation.

The shift from periodic pentesting to continuous validation requires more than automation. It requires a system capable of executing realistic attack paths, producing reproducible evidence, and repeating validation as systems evolve.

KEY MESSAGE

Infiltra operationalises agentic testing to deliver continuous, evidence-based security validation.

Infiltra applies an agentic testing framework to enable this model. Autonomous agents interact with applications using browsers, APIs, and workflow-aware testing logic to validate vulnerabilities across real user flows and system behaviours.

Rather than producing large volumes of speculative alerts, the platform focuses on confirming exploitable conditions and generating reproducible evidence that security teams and engineering teams can act on with confidence.

Because testing can run continuously, organisations can verify whether vulnerabilities remain present as applications change, confirm that fixes remain effective, and maintain ongoing visibility into their security posture.

Infiltra also supports integration with development workflows, enabling validation results to inform release decisions, remediation prioritisation, and governance reporting. This allows security testing to function not only as an assessment activity, but as a repeatable control within the software delivery lifecycle.

Through this model, organisations can move beyond point-in-time pentesting and establish continuous validation of application security risk.

Contact us for more information — Email: contact@infiltra.ai

Continuous validation requires more than scanning. It requires systems capable of executing, verifying, and repeating realistic attack paths.

References

Selected references used for quoted statistics, framework references, and supporting claims in this white paper.

1. **Google Cloud / DORA Research.** Accelerate / State of DevOps materials referencing elite delivery performance and 973× higher deployment frequency. Available at: https://cloud.google.com/resources/state-of-devops
2. **Systems Sciences / remediation cost timing research.** Widely cited research on the increasing cost of fixing defects and vulnerabilities later in the lifecycle. Commonly referenced through NIST-style and Systems Sciences Institute summaries.
3. **OWASP API Security Top 10 — 2023 Edition.** Available at: https://owasp.org/API-Security/editions/2023/en/0x11-t10/
4. **OWASP API1:2023 — Broken Object Level Authorization.** Available at: https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/
5. **OWASP API2:2023 — Broken Authentication.** Available at: https://owasp.org/API-Security/editions/2023/en/0xa2-broken-authentication/