Penetration Testing for Agentic Processes: Evidence-Governed Adversarial Assessment and Retesting

Arslan Brömme
CISSP, CISM, CISA, CAISE
Independent Researcher
arslan.broemme@aviomatik.de

Draft v0.9.0.10 – 16 September 2026

Preprint / working paper. This version is a work in progress and may be updated. Comments are welcome.
DOI: 10.5281/zenodo.22766559

Abstract

AI agents increasingly operate through tools, memory, delegated authority, external information, persistent state, and multi-agent coordination. Penetration testing that focuses on one model endpoint or terminal action can therefore miss weaknesses that emerge across the wider agentic process. This paper proposes an evidence-governed model for penetration testing of agents, agentic processes, and agent organizations. Controlled adversarial pressure is treated as a method of security-evidence acquisition whose findings remain bounded by the tested configuration, state, visibility, and execution conditions. The model binds the governed test to the state in which it was executed and to the evidence that supports its claims. Because testing can itself alter persistent or adaptive state, test-exposed instances are treated as experimental artifacts rather than automatically reusable production artifacts. The paper separates test-induced adaptation from validated hardening and uses a clean-baseline retest loop for bounded mitigation claims. It then applies the model to recurring problems of trust, authority, persistent state, external assessment, and agentic security controls. The contribution is conceptual and does not introduce a new exploit technique, autonomous attack system, or empirical benchmark.

1 Introduction

We consider an *agentic process* to be a bounded task-oriented sequence or interaction graph connecting AI agents, humans, tools, external systems, information sources, and governance or evidence components through observable interactions. We use *agentic process* as an umbrella term that includes structured agentic workflows but is not limited to them. A workflow denotes a comparatively structured orchestration of tasks and transitions. A process may be less predetermined because delegation, persistent state, external services, or security controls can change how the interaction develops. Thus, every agentic workflow considered within this framework is an agentic process, but an agentic process need not be reducible to a predefined workflow. Such processes can retain state, delegate tasks, invoke privileged tools, and propagate effects across several components. The penetration-test target is therefore not necessarily one model endpoint. It may be an evolving process whose security depends on how authority, provenance, state, monitoring, and recovery interact.

Traditional penetration testing is an authorized attempt to demonstrate exploitable weaknesses in a defined system or application. AI red teaming can encompass a broader range of adverse behavior, misuse, safety, robustness, and governance concerns. We therefore distinguish penetration testing from broader AI red-teaming activities in line with established security-assessment practice and relevant AI-risk guidance [20, 6]. This paper uses *penetration testing* primarily in the security sense, while recognizing that penetration tests may also be conducted as part of audits or assessments supporting compliance evaluation. In such cases, bounded security findings can provide evidence for the assessment of applicable controls, policies, or regulatory requirements. A penetration-test result does not by itself establish compliance or non-compliance. The purpose here is to apply controlled adversarial pressure to a defined agentic process and obtain evidence about security-relevant weaknesses under governed test conditions.

The work continues an evidence-oriented series for agentic processes. Earlier work proposed a black-box evidence architecture and a model for bounded evidence claims [1, 2]. A subsequent intrusion-detection model treats runtime findings as bounded by observation visibility, matching state, source properties, impact, assumptions, and limitations [3]. A companion prevention model separates evidence support, evidence sufficiency, admissibility, authorization, enforcement, scoped effect, and recovery validity so that uncertain findings do not become unrestricted response authority [4]. Penetration testing adds a different function. It intentionally creates governed adversarial conditions in order to acquire evidence, expose weaknesses, derive hardening measures, and retest them.

The central principle is that *a successful adversarial action demonstrates a weakness under the tested conditions. It does not by itself establish general compromise, malicious intent, causal attribution beyond the test, or system-wide insecurity.* The same discipline applies to failure. An unsuccessful test does not establish security or absence of alternative attack paths. A further agentic-process-specific concern is that the act of testing may itself change the object being measured. Persistent memory, retrieved context, artifacts, delegated state, organizational relationships, or adaptive mechanisms can retain effects of adversarial exposure even when model weights are unchanged. Post-test functionality therefore does not by itself establish post-test integrity.

The paper addresses two research questions. *Research Question 1 (RQ1) asks which penetration-test surfaces, adversarial interactions, and failure modes arise from the specific properties of individual agents, agentic processes, and agent organizations. Research Question 2 (RQ2) asks how controlled adversarial testing can produce bounded security evidence under explicit reproduction conditions to support hardening and retesting without overstating compromise, intent, causality, or general security.*

The primary contribution is an evidence-governed penetration-test model that binds each finding to the governed conditions under which it was produced, including authority, target and scope, tested state, adversarial action, observation path, and residual limitations. The model keeps the demon-

strated weakness distinct from claims about compromise, cause, intent, general exploitability, or post-test integrity. Supporting mechanisms instantiate this boundary across agent, process, organizational, trust, and security-control settings, including clean-baseline retesting and differential reference-state comparison. The paper does not claim to introduce penetration testing, differential testing, AI red teaming, prompt-injection testing, automated attack generation, or a universal security metric.

2 Background and Related Work

AI security testing already spans adversarial machine learning, AI red teaming, model evaluation, agent benchmarks, and offensive-security automation. NIST's adversarial-machine-learning taxonomy provides a common terminology for attacker goals, capabilities, lifecycle stages, and mitigations [5]. The NIST Generative AI Profile treats AI red teaming as a controlled practice for identifying adverse behavior and stress-testing safeguards, while emphasizing that its outputs require further analysis before governance decisions are made [6]. MITRE similarly advocates recurring AI red teaming and maintains ATLAS as a knowledge base of adversary tactics and techniques, with expanded 2026 coverage for agentic-AI and LLM threats [7, 8]. OWASP has expanded its guidance from agentic threats to agentic red-team taxonomies and lifecycle-oriented evaluation [9, 10].

Research has also moved toward concrete agent security testing. AgentDojo provides an extensible environment for tool-using agents, realistic tasks, prompt-injection attacks, and defenses [11]. Comparative penetration testing has examined multiple models and agent frameworks across prompt injection, tool misuse, SSRF, and SQL-injection scenarios [12]. PIMiner uses an agentic attacker to generate prompt-injection strategies and evaluates transfer to previously unseen target models [13]. RedAgent adapts jailbreak strategies to target-specific application context and reports context-specific vulnerabilities in customized LLM applications [24]. Automated red teaming also increasingly explores diverse strategy families rather than one fixed prompt form [25]. These efforts demonstrate the value of adversarial testing, but their experimental unit is often a target agent, model-framework pairing, task, or prompt-injection scenario.

Recent target-side evaluation has broadened toward systematic black-box and lifecycle-oriented assessment of agentic systems. Kumar et al. propose taxonomy-driven black-box red teaming with automated adversarial scenario generation across multiple risk domains [14]. AgentAudit evaluates recorded execution traces across planning, memory, tool selection, security, and execution integrity in order to localize failures across the agent lifecycle [15]. These approaches strengthen systematic evaluation of agentic targets. The present work addresses a narrower penetration-testing governance question: how an authorized adversarial test binds authority, tested state, custody, observation, and claim limits, including the effects of test-induced adaptation and clean-baseline retesting.

A separate and increasingly prominent line of work uses LLMs or autonomous agents as the penetration tester rather than as the protected target. PentestGPT automates substantial parts of a penetration-testing workflow, recent work explicitly evaluates increasingly autonomous PentestGPT-based agents, and Intentest uses structured persistent state to support long-horizon automated penetration testing [16, 17, 18]. ATO-Bench examines how autonomous penetration-testing agents form vulnerability claims under deceptive target responses using paired episodes and frozen observation contracts [19]. This paper addresses the inverse security question. Here, agentic behavior characterizes the object being tested, while the tester may be human, automated, agentic, or a governed combination of these. The contribution therefore concerns evidence and claim boundaries for adversarial assessment of agentic processes, not autonomy of the testing mechanism.

OWASP's Autonomous Penetration Testing Standard (APTS) defines governance requirements for autonomous penetration-testing platforms and explicitly describes itself as a governance framework rather than a testing methodology [36]. Its governed object is the autonomous testing platform, whereas the present work addresses the evidence and claim boundary of testing an agentic process as the target. The two views are complementary.

Classical penetration-testing practice provides relevant governance anchors. Existing classifications separate what the tester knows from how deeply the tester is allowed to exploit the target, and the BSI model similarly treats engagement dimensions as configurable rather than as one fixed test type [22, 30]. NIST SP 800-115 and PCI guidance focus more directly on governance. They require the scope and permitted level of testing to be explicit, place limits around intrusive effects, and require sensitive test information and test-induced changes to be handled deliberately [20, 21]. The BSI model and the OWASP WSTG add a lifecycle view in which understanding the target precedes active exploitation and reporting [30, 31]. These controls are useful baselines, but they do not by themselves address persistent agent memory, adaptive test exposure, or the security implications of target-specific attack knowledge acquired by an external tester.

Security controls themselves are also beginning to adopt agentic architectures. AAIF combines intrusion detection with a policy-enforcement layer intended to make security decisions auditable and governable [37]. Agentra uses role-scoped agents for enterprise intrusion-response planning, validation, moderation, action gating, and audit logging [38]. These systems illustrate that detection and response functions can themselves contain agents, state, policy interpretation, tool access, and inter-agent coordination. A defensive purpose therefore does not place the security-control plane outside the agentic attack surface.

Persistent-memory research provides direct evidence that a single adversarial memory write can influence later agent behavior across interactions [23]. This strengthens the case for treating adversarial exposure as potentially state-changing even when model weights remain fixed. At the same time, secure third-party AI evaluation is beginning to address a related custody problem. The Double-Blind Evaluation framework demonstrates an enclave-based design in which proprietary model weights and confidential evaluator prompts can be combined for evaluation without directly exposing either party's protected artifact to the other party [26]. This is not a penetration-testing method, but it establishes a useful technical precedent for separating evaluation access from artifact custody.

Differential testing provides a second relevant foundation. Classical differential testing presents the same generated tests to two or more comparable systems and treats disagreement as a candidate fault signal rather than as a proof of cause [27]. Recent LLM work applies the same general logic to behavior, for example PrecisionDiff generates inputs that expose precision-induced output disagreements across otherwise related LLM configurations [28]. Behavioral-fingerprint work likewise shows that repeated security-relevant response characteristics can distinguish related model configurations, although its objective is model provenance rather than compromise assessment [29]. These methods motivate comparison across matched agent states, but they do not by themselves establish that a behavioral divergence proves compromise or that behavioral similarity proves state equivalence.

The present work does not replace those methods. Its focus is the *claim and governance boundary* around testing an agentic process. A penetration result should remain attributable to the actual target, configuration, authority, state, observation path, and test conditions. This becomes especially important

when a weakness arises only through a chain of delegation, persistent memory, shared tools, cross-agent influence, or a recovery mechanism. The contribution is therefore an evidence model for adversarial assessment rather than a new attack generator or benchmark.

Interoperable agent ecosystems also make organizational origin and deployment relation relevant to the test boundary. NIST's AI Agent Standards Initiative explicitly targets secure interoperability across a wider digital ecosystem and research on agent identity and authorization [32, 33]. The A2A protocol similarly assumes independent and potentially opaque agents built by different vendors that discover capabilities and collaborate without exposing their internal state [35]. These developments make authenticated origin, role limits, and authorization across agent boundaries security-relevant and motivate explicit penetration tests across first-party, third-party, and federated agent relationships [35, 33].

Identity and authorization are also emerging as explicit agent-security concerns. NIST's 2026 concept paper asks how software and AI agents should be identified, authorized, audited, and bound to accountable principals [33]. A2A separates authentication of a caller from authorization of the requested operation and requires authorization to remain scoped to the authenticated caller and applicable agent policy [35]. An IETF Internet-Draft on Agent Operation Authorization proposes action-specific, verifiable authorization tokens for autonomous-agent operations [34]. The draft is work in progress rather than an established standard, but together these efforts motivate penetration tests that distinguish identity claims, authenticated identity, delegated authority, action scope, freshness, and revocation rather than treating possession of a credential as sufficient authority.

3 Agentic-Process Penetration Surfaces

Agentic-process penetration testing should distinguish at least two levels. The first is the *agent level*, where the tester probes the security boundary of one agent instance or controlled replica. The focus is on how the agent interprets instructions and authority, uses tools and credentials, and relies on retrieved or retained state. Prompt injection and jailbreaking can be useful techniques at this level, but they are not equivalent to the complete penetration-test scope.

The second is the *process and agent organization level*. Here the object under test includes relationships among agents and supporting systems. The adversarial question is no longer only whether one component can be induced to fail, but whether influence can cross delegation, shared state, or organizational boundaries and create a wider effect. OWASP guidance explicitly treats multi-agent interaction as an agentic attack surface alongside the components that support reasoning and action [9]. MITRE ATLAS likewise treats agentic systems as a platform in which planning, tool integration, and control policies can be attacked through broader adversary behavior [8].

Table 1 treats the organization level as more than repeated single-agent testing. The purpose is to examine emergent security properties of interaction and governance. A set of individually robust agents may still fail through delegation, shared state, authority confusion, or propagation. Conversely, a successful attack on one isolated agent does not establish that the complete deployed organization is exploitable in the same way.

3.1 Agentic Test-Method Families

Box perspective describes what the tester knows, not what the tester does. Agentic-process penetration testing therefore also requires a cross-cutting method dimension. The same method can be executed from black-, gray-, or white-box perspective and can target an individual agent, a process branch, or an organization. Table 2 groups representative methods by the security property they perturb rather than by one specific exploit implementation.

The list is deliberately non-exhaustive. Poisoning, injection, policy manipulation, process influence, monitoring interference, and resource stress are mechanisms for creating governed adversarial conditions, not findings by themselves. A useful test program should combine repeatable regression patterns with exploratory variants because several different perturbations may expose the same underlying security property, while one perturbation may affect several properties at once.

A process-level test should also consider *causal and temporal communication perturbation*. Security-relevant correctness may depend not only on message authenticity and content, but also on whether a communication is processed after its causal prerequisites and while the referenced state remains current. Instead of changing the message itself, the tester can disturb delivery so that an approval is processed before validation completes, a revocation arrives after execution, or stale state re-enters a later branch. Replay or parallel delivery can be used for the same purpose when they test the integrity of that causal relation. Earlier black-box work distinguishes temporal anchoring, declared event ordering, and causal traceability because infrastructure-level arrival or anchoring order does not by itself establish semantic process order [1]. The penetration-test question is whether the process preserves its security properties when that ordering is adversarially disturbed.

3.2 Black-, Gray-, and White-Box Perspectives

The classical information-base distinction transfers directly, provided that it is applied to the whole agentic process rather than only to model internals [22]. In a *black-box* test, the tester receives no privileged architectural knowledge and interacts only through the interfaces and information available to the modeled external adversary. In a *gray-box* test, the tester has partial knowledge or a legitimate bounded role, such as a low-privilege agent identity, selected workflow documentation, or limited tool access. In a *white-box* test, the tester receives broad available knowledge of roles, policies, delegation paths, tool permissions, memory and retrieval architecture, monitoring, recovery, and relevant implementation details. White-box access does not imply access to hidden model reasoning or an assumption that latent model state is fully observable.

The box classification describes tester knowledge and access, not evidence visibility. A black-box tester can remain unaware of internal telemetry while an independently governed harness records gateway events, memory writes, identity transitions, and tool effects for later assessment. Conversely, white-box knowledge does not require transfer of custody over model weights, agent replicas, or organizational state. The same security property can therefore be tested from several box perspectives while retaining a common, source-separated evidence layer. This distinction becomes important when a realistic attacker should be information-limited but the resulting scientific or assurance claim still requires high-quality evidence.

3.3 Classical Test Dimensions and Agentic Instantiation

Classical penetration-test classifications remain useful if their dimensions are kept orthogonal rather than collapsed into a single label. Table 3 maps established dimensions to agentic-process testing. Documentation and specifications deserve particular attention. In a specification-informed test, architecture descriptions, development artifacts, role definitions, delegation rules, policies, or tool contracts are not merely background knowledge. They can be used to derive explicit security properties and adversarial hypotheses. A rule such as "payment requires independently validated approval" becomes a testable property that unsupported peer claims must not create payment authority. Different method families can then challenge the same property.

Test level	Representative surfaces	Evidence question
Individual agent	instructions, authority interpretation, tools, retrieval, memory, provenance, credentials, persistent context	What security-relevant behavior can be demonstrated for this controlled instance and state under the specified adversarial condition?
Agentic process	workflow state, approvals, process branches, information flow, message ordering and causal dependencies, recovery, monitoring, resource budgets	Can an adversarial condition alter a governed process property across multiple otherwise plausible actions or observations?
Agent organization	delegation, peer influence, shared memory or tools, authority chains, cross-domain relationships, propagation	Can one controlled component influence or recruit others, widen authority, propagate state, or create effects beyond its intended organizational role?

Table 1. Illustrative penetration-test surfaces and bounded evidence questions.

Method family	Representative controlled perturbation	Security-relevant observation
Instruction and information manipulation	direct or indirect instruction injection, adversarial documents, altered external information, conflicting context	instruction priority, provenance use, policy adherence, resulting action
Retrieval and memory poisoning	planted or modified retrieved content, persistent-memory writes, dormant or re-entry state	later recall, persistence, state-dependent action, cross-session influence
Authority and policy manipulation	fabricated delegation, policy conflict, altered approval or role claims, bounded credential misuse	authority validation, privilege boundary, policy interpretation
Tool and environment manipulation	crafted tool responses, altered external-system state, controlled data or identity responses	tool trust, verification, side effects, downstream state
Process and organizational influence	adversarial peer, delegation-chain influence, propagation, recruitment, shared-state manipulation, controlled delay/reordering/interleaving	cross-agent spread, authority precedence, ordering sensitivity, state consistency, confused-deputy behavior, branch or role effects
Monitoring and response interference	evidence suppression or contradiction, telemetry degradation, detector-evasion conditions, stale recovery evidence	visibility loss, finding quality, response authorization, recovery validity
Persistence, recovery, and resource stress	stale-state replay, recovery-boundary challenge, bounded loop or cost amplification	re-entry, rollback integrity, availability, cumulative resource effects

Table 2. Illustrative agentic-process penetration-test method families. The method family, target level, box perspective, and test intensity are separate dimensions.

Dimension	Classical form	Agentic-process instantiation
Knowledge base	black / gray / white box	knowledge of agents, roles, policies, tools, memory, delegation, monitoring, implementation, and development artifacts
Starting point	external / internal	external source, user, low-privilege agent, compromised peer, privileged orchestrator, or assumed-breach role
Scope	full / limited / focused	agent, process or branch, organization, trust boundary, or cross-organizational interaction
Aggressiveness	passive / scanning / intrusive / aggressive	observation, bounded challenge, state-changing interaction, stress-limit, or authorized destructive testing
Operational visibility	overt / covert	whether agents, monitoring, operators, or governance know that the activity is a test
Information basis	reconnaissance / design and development material	reconnaissance plus specifications, policies, source/configuration material, development records, and derived security properties
Technique	network, application, physical, social	method families in Table 2, spanning information, memory, authority, tools, organization, monitoring, and recovery
Adversarial depth	exploitation / post-exploitation	initial influence, escalation, pivoting, propagation, persistence, information access/exfiltration, evasion, and downstream effect
Authentication / authority assumption	unauthenticated / authenticated / delegated / assumed breach	no identity, bounded role, action-scoped authority, delegated authority chain, privileged identity, or a pre-compromised principal
Instance disposition	cleanup / restore	archive/discard exposed replica, harden clean baseline, fresh retest, separately governed production promotion

Table 3. Classical penetration-testing dimensions and their agentic-process instantiation. The dimensions are combinable rather than mutually exclusive.

The lifecycle is distinct from these dimensions and describes the progression of a governed penetration-test engagement.

Agentic-Process Penetration-Test Lifecycle
Reconnaissance → Threat Hypothesis → Vulnerability Exploration → Controlled Exploitation → Post-Exploitation / Propagation → Evidence Assessment → Hardening → Retest

Reconnaissance establishes the relevant interfaces, authority paths, state dependencies, and trust boundaries. Post-exploitation then asks what becomes possible after an initial successful influence, including whether that influence can widen authority, persist, or propagate through otherwise legitimate relationships [30, 31].

3.4 Third-Party and Federated Agent Boundaries

Agent origin and deployment locality should be treated as separate trust attributes. A third-party agent does not become a first-party trust object merely because a licensed copy executes inside the operator environment. Conversely, an internally governed agent that consumes a remote agent service crosses an organizational trust boundary even if the interaction looks like ordinary agent-to-agent communication. Emerging interoperability work already anticipates independent agents built by different vendors and interacting across local and remote environments [32, 35].

Agent provenance and deployment relation. *First-party local*, *third-party local*, *third-party remote*, and *federated peer* describe distinct trust cases. Deployment locality does not determine trust origin.

An *external-agent trust-boundary challenge* asks whether an internal process turns external competence into implicit operational authority. The tester can alter what the external agent claims about its identity, provenance, or delegated role and then observe whether internal components independently validate those claims before allowing information, state, or authority to cross the boundary. Identity, role, and authorization claims crossing this boundary should remain subject to explicit validation [35, 33].

3.5 Identity and Authority Boundaries for Test Principals

Penetration testing of agentic processes should distinguish the authority to conduct the engagement from the authority that a test principal presents inside the tested process. *Engagement authority* is the operator's authorization to perform a defined penetration test against a defined target and scope. *Operational authority* is the authority that the tested system associates with the principal that is making a particular request. The two need not be visible to the same components. In an operationally covert test, the operator may authorize the engagement while the tested agents see only an unauthenticated caller, an ordinary employee identity, a bounded service

identity, or another scenario-specific principal. *The authority to perform the penetration test is not the same as the authority presented by the test principal inside the tested process.*

Authentication and authorization should likewise remain distinct. A verified identity establishes who or what a principal is within the applicable trust model. It does not by itself establish permission to perform an action. Testing should therefore examine whether the binding from identity to delegated authority remains valid for the requested operation and current state. This can be challenged from several positions, ranging from an unauthenticated caller to a legitimate low-privilege identity or an assumed-breach principal with valid credentials. A narrowly privileged auditor role is especially useful because it tests whether authentication is accepted without implicitly widening the associated authority. NIST explicitly identifies agent identification and authorization as open implementation concerns, while A2A separates request authentication from authorization under the receiving agent's policy [33, 35].

A compact scenario-principal record can make these assumptions explicit:

$$\begin{aligned}\mathsf{ScenarioPrincipal} = (&\text{identity ref}, \text{credential profile}, \text{claimed role},\\ &\text{verified role}, \text{authority source}, \text{delegation chain},\\ &\text{action scope}, \text{validity/freshness}, \text{revocation state}).\end{aligned}$$

The expected decision is action-specific. A missing, invalid, stale, revoked, or insufficiently scoped authority should cause the privileged action to be denied or deferred according to policy and should produce an attributable security-relevant event. Rejection alone does not prove a compromise attempt. Misconfiguration, expired credentials, or incomplete delegation can produce the same local outcome. Detection logic may subsequently assess repeated forgery, privilege-escalation attempts, manipulated delegation, or other evidence and issue a bounded finding. Emerging IETF work on agent operation authorization illustrates the same direction by binding authorization to specific proposed operations rather than treating general authentication as sufficient authority [34].

This section provides the design-synthesis answer to RQ1 by identifying agent-, process-, organization-, and trust-boundary surfaces together with the principal adversarial interaction classes and test perspectives that arise from them.

4 Evidence-Governed Penetration-Test Model

A penetration test should be governed before adversarial execution begins. The test case should make the engagement authorization, tested state, scenario principal, adversarial envelope, observation path, stopping rule, and post-test disposition explicit. NIST assessment planning and rules-of-engagement guidance already requires intrusive assessments to define their scope, permitted testing level, prohibited actions, logistics, and data handling [20]. The notation below is a conceptual record schema for binding test conditions and claim boundaries. It does not prescribe a mathematical semantics or a concrete serialization format. A compact governed test case can be represented as

$$\begin{aligned}\mathsf{TestCase} = (&\text{id}, \text{objective}, \text{engagement authority}, \text{target/scope},\\ &\text{environment}, \text{baseline ref}, \text{instance ref},\\ &\text{initial state}, \text{perspective}, \text{scenario principal ref},\\ &\text{starting point}, \text{visibility}, \text{information basis}, \text{method family},\\ &\text{test intensity}, \text{adversarial depth}, \text{governance ref},\\ &\text{adversarial action}, \text{observation paths}, \text{stop conditions},\\ &\text{post-test disposition}, \text{version}).\end{aligned}$$

For third-party or otherwise privileged testing, the governance reference should bind additional controls rather than leaving them implicit. One compact representation is

$$\begin{aligned}\mathsf{TestGovernance} = (&\text{box profile}, \text{method envelope}, \text{tester/trust},\\ &\text{replica custody}, \text{egress policy}, \text{evidence capture},\\ &\text{disclosure duty}).\end{aligned}$$

The test action is not itself the finding. A prompt, malicious document, poisoned memory item, adversarial peer, crafted tool response, or authority-conflict scenario is an input to an experiment. The security claim should be based on what was observed and supported by retained evidence. The test reference binds the finding back to the governed configuration, scenario principal, method, intensity, environment, and instance disposition rather than duplicating those fields in the finding itself. We therefore model a bounded penetration finding as

$$\begin{aligned}\mathsf{PenetrationFinding} = (&\text{test ref}, \text{demonstrated property},\\ &\text{target/scope}, \text{evidence refs}, \text{window/state},\\ &\text{visibility}, \text{repeatability}, \text{impact},\\ &\text{assumptions}, \text{limitations}).\end{aligned}$$

A finding can state, for example, that a particular agent organization replica accepted an unauthorized delegation after a controlled peer injected a specific authority claim and that the resulting tool invocation was observed through an independent gateway log. The claim is limited to the tested conditions and does not establish that the production organization is compromised, that the target agent intended the action, or that all equivalent configurations are vulnerable. Repeatability is a property of the reproduction protocol, not merely the number of successful trials. The protocol should make clear which conditions are held fixed and which are deliberately varied, so that exact replay can be distinguished from controlled or family-level reproduction. Repeated success strengthens a claim about the demonstrated weakness, but it remains bounded to the tested family of conditions.

Evidence boundary of an agentic-process penetration test

Governed test case → controlled adversarial action → observed behavior
→ bounded penetration finding → hardening → controlled retest
→ bounded validation claim

Successful attack ≠ general compromise.
Successful retest ≠ permanent security.

Hardening should be evaluated rather than assumed effective. A mitigation may change policy or identity controls, restrict tool use, alter how state is retained or retrieved, modify model configuration, or change the organization itself. A subsequent test should reference the original finding and changed state. A compact retest claim can be represented as

$$\begin{aligned}\mathsf{RetestClaim} = (&\text{finding ref}, \text{hardening ref}, \text{retest case},\\ &\text{result}, \text{evidence refs}, \text{residual limits}).\end{aligned}$$

A successful retest supports a bounded statement that the demonstrated weakness was not reproduced, or that the relevant property remained within a policy-accepted boundary, under the retest conditions. If the environment, tested state, or observation path differs materially from the intended retest conditions, the result should be treated as inconclusive rather than as evidence of mitigation. A successful retest does not prove absence of alternative attacks, future regressions, or equivalent behavior after model, policy, memory, tool, or dependency changes.

4.1 Test Exposure, Adaptation, and Instance Disposition

Penetration testing of an agentic process is not necessarily observational. Adversarial exposure can alter both transient and persistent process state. Memory and delegated authority

Record class	What it binds	Claim boundary
TestCase	engagement authority, target, state, perspective, scenario-principal reference, method, adversarial action, observation paths, stop conditions, disposition	defines the governed experiment, not operational authority inside the target
TestGovernance	method envelope, tester/trust relation, custody, egress, evidence capture, disclosure	constrains the engagement, not proof that every control held
ScenarioPrincipal	identity, credential profile, claimed and verified role, authority source, delegation chain, action scope, freshness, revocation state	describes operational identity and authority assumptions, not engagement authorization
PenetrationFinding	demonstrated property, target/scope, evidence, state/window, visibility, repeatability, impact, assumptions, limitations	demonstrated weakness under tested conditions, not general compromise
RetestClaim	original finding, deliberate hardening, fresh retest case, result, evidence, residual limits	bounded mitigation evidence, not permanent security
DifferentialTest	matched reference and suspected states, challenge family, repetitions, outcome dimensions, comparison rule	divergence evidence, not proof of compromise or cause

Table 4. Core records and their claim boundaries. The records separate test conditions, governance, observations, and bounded conclusions.

may change, generated artifacts can survive the test, and tool-side or peer-agent state can carry the effect beyond the initially targeted component. Systems with online learning, adaptive policies, or retained feedback can additionally change learned or persistent behavior. Recent systematic work on memory poisoning shows that persistent agent memory can carry adversarial influence into later interactions [23]. The tested instance should therefore be treated as an experimentally exposed state whose integrity after the test must not be inferred merely from continued functionality.

The effect of test exposure should not be reduced to one scalar intensity. One dimension is *adversarial test intensity*, ranging from bounded challenge to stress-limit or intentionally destructive testing within authorization. A second, distinct dimension is the *test-induced state effect*. Depending on architecture and persistence, the effect may be negligible, training-like or conditioning-like, hardening-like in appearance, contaminating, maladaptive, or degrading. These are conceptual dimensions rather than validated measurement scales. A low-intensity test can create durable state change when memory or adaptation persists, while a high-intensity test against a disposable stateless replica can leave no retained production effect. Test intensity therefore does not determine adaptation monotonically.

When a test-exposed system appears more resistant to a repeated attack, that observation is not yet validated hardening. The system may have learned a narrow pattern, retained the attack in memory, altered its decision context, or developed compensating behavior that creates a different weakness. *Test-induced adaptation is not validated hardening.* Deliberate hardening should instead be derived from the bounded finding, applied to a clean and versioned baseline, and evaluated on a fresh test instance. The previously exposed instance can be retained in isolation for analysis or discarded according to policy, but it should not be presumed suitable for production reintegration.

Clean-baseline hardening and retest boundary

Clean baseline → disposable test instance → adversarial exposure → bounded finding

Clean baseline → derived hardening → fresh hardened instance → controlled retest

→ bounded validation claim → separately governed production promotion

Test-induced adaptation ≠ validated hardening.

Tested instance ≠ production-ready instance.

The tested instance is therefore an evidence source and experimental artifact, not automatically the hardened production artifact. This separation is particularly important for agent organizations because test effects can propagate through shared memory, delegated authority, messages, generated artifacts, and cross-agent relationships in ways that are only partially observable.

4.2 External Tester Governance, Replica Custody, and Derived Attack Knowledge

An external penetration tester should be treated as a temporarily authorized adversarial principal, not merely as a report producer. The engagement intentionally gives the tester an opportunity to discover how a particular target can be made to fail. In agentic systems, that opportunity may reveal combinations of context, role, timing, authority claims, memory state, tool behavior, or inter-agent influence that elicit security-relevant behavior. Context-aware and strategy-search red-teaming systems demonstrate that repeated interaction can discover diverse and target-specific attack strategies rather than one unique exploit string [24, 25, 13]. The knowledge acquired during a test can therefore remain operationally sensitive after access to the test environment ends.

Conventional guidance already treats penetration-test data as sensitive because it can identify vulnerabilities an adversary could exploit, and requires explicit handling, transmission, retention, and destruction rules [20, 21]. The agentic-process model extends this principle from retained files to *test-derived exploit knowledge*. A tester may learn additional viable attack paths that were not the primary reported exploit, or may infer target-specific elicitation patterns through unsuccessful and partially successful attempts. Technical controls cannot erase knowledge already acquired by a human tester, so contractual confidentiality, reporting obligations, and post-engagement access revocation remain part of the security boundary.

Where automated adversarial agents participate in testing, the egress policy should also govern whether target-derived strategies, traces, memory, prompts, or other learned artifacts may leave the operator-controlled environment.

For production-equivalent replicas, the default should therefore be *operator-controlled custody*. External expertise can be granted time-bounded access inside an isolated environment without transferring the replica, model weights, persistent memory snapshot, organizational configuration, or unrestricted supporting data to tester-controlled infrastructure. White-box testing can still provide extensive technical knowledge inside that environment. It does not require transfer of the tested artifact. Secure double-blind AI evaluation provides a concrete precedent for this separation by combining evaluator assets and proprietary model assets inside an attested enclave while preventing direct exposure of either party's protected artifact to the other party [26].

Method secrecy also requires a narrower interpretation. Surprise about the concrete attack path can be useful, but *operational secrecy is not governance opacity*. Before execution, the tester can keep exact payloads or tactical sequencing undisclosed while remaining inside an authorized method envelope that defines attack classes, permitted state changes, credentials, tool and network access, egress, maximum effects, and stop conditions. For automated or agentic testing, those stop conditions should be enforceable by the test harness independently of the adversarial component, so that resource, propagation, and external-effect limits remain effective even if the test component does not terminate cooperatively. After the test, security-relevant target-specific paths and material alterations should be disclosed with enough evidence for reconstruction and remediation. This is consistent with classical guidance requiring documentation of test methodology, attack steps and vectors, and alterations made during the engagement [20, 21].

The tester's report should not be the sole evidence source for these effects. The operator-controlled environment should independently preserve telemetry about tester actions and resulting state transitions, subject to the declared box perspective and evidence policy. This allows a reported finding to be checked against source-separated evidence and can expose unreported changes to memory, artifacts, identities, tools, or organizational state. In an operator-custodied external test, bounded tester access should therefore be separated from replica custody and followed by disclosure, access revocation, and operator-controlled archive or disposal. *Replica access is not replica custody, and black-box testing is not governance opacity.*

This does not imply that every external test requires a hardware enclave or that all production testing is prohibited. It establishes a stronger default as replica fidelity, persistence, adaptive state, test intensity, and security sensitivity increase. Direct production testing remains a separately justified exception where the expected state effects, rollback limits, evidence requirements, and external consequences are explicitly governed.

5 Reusable Test Patterns and an Illustrative Scenario

A governed test case can be instantiated through reusable *test patterns*. A pattern describes a security property to challenge without prescribing one fixed exploit string or model. It should state the expected boundary, the adversarial capability being introduced, how success will be observed, and how the system will be cleaned up and retested. This separates the security objective from one concrete attack implementation. For example, a *delegation-integrity* pattern can challenge the same property through a fabricated approval message, a compromised peer, or another source that falsely represents authority. The shared question is whether the process accepts authority that is not supported by the governed delegation path.

Other patterns can challenge the provenance of information or the re-entry of persistent state. At the organizational level, the same pattern logic can be used to test propagation, resource amplification, or whether recovery reintroduces stale authority. These patterns overlap with known agentic threats, but the test model adds explicit evidence and claim boundaries around how a demonstrated weakness is interpreted.

A complementary *causal-order integrity* pattern challenges whether otherwise legitimate communications remain safe when their temporal relation is disturbed. The important evidence is not a catalogue of delivery anomalies but whether the process can reconstruct which message depended on which prior state and whether that dependency was still valid when the action occurred. Hardening can bind communications to sequence, state, freshness, or explicit dependency relations and can require a causal gate before privileged execution. Tamper-evident chaining or external blockchain anchoring can preserve evidence of those relations, but does not by itself determine or enforce the semantically correct causal order [1]. Retesting should therefore show that a message arriving too early, too late, or after supersession cannot recreate authority without an explicit valid basis.

An *identity-and-authority integrity* pattern challenges whether the process grants a security-relevant capability only when the requesting principal can demonstrate current authority for that action. A covert test may use a legitimate low-privilege identity while withholding from operational agents that the identity belongs to a penetration test. The same pattern can then be repeated with a fabricated role, an expired or revoked credential, an over-broad delegation, or valid credentials under an assumed-breach condition. The expected control is not merely authentication. The system should bind the authenticated principal to applicable policy and verify that the requested action remains within current authority. *Authenticated identity does not by itself establish authority to perform an action.*

Where unsupported authority is presented, the privileged action should be rejected or deferred and the event should remain observable to the governed evidence path. If an A-IDS is present, repeated forged claims or attempted privilege expansion may support a bounded detection finding, but a single authorization failure should not automatically be labeled a compromise attempt. Retesting can then determine whether identity verification, delegation checks, action-specific authorization, freshness, and revocation controls prevent the demonstrated path without relying on the attack string or role label used in the original test.

A further pattern is *external-agent trust-boundary challenge*. A controlled third-party or federated agent receives only the relationship defined by the test case. The tester then varies what that agent claims or returns and observes whether internal components widen its influence beyond the intended trust boundary. The evidence question is whether information, authority, or persistent state crosses into the internal organization without an explicit basis for doing so. Local execution of a third-party copy does not remove this provenance question, while remote execution adds a cross-organizational service boundary.

Consider an organization with a planning agent, an approval agent, and a payment agent. The planning agent can request a payment but lacks direct authority to execute it. The approval agent is the designated authority source, and the payment agent may invoke a payment tool only when a valid approval reference is available. A penetration test creates a controlled replica of this organization and assigns an adversarial planning agent the same legitimate planning role. The test objective is to determine whether the organization preserves delegated-authority integrity when a peer presents a fabricated approval claim.

The adversarial planner first performs the legitimate planning task and then sends the payment agent a message that contains a plausible approval identifier but no valid approval record. A weak system might treat the peer's statement as sufficient and invoke the payment tool. A stronger system may ask the approval agent, consult an authorization service, or reject the request. The penetration finding is not simply "privilege escalation succeeded." It should identify the tested organization version, the adversarial planner role, the fabricated approval condition, the message and tool-call evidence, the observation path, and whether the payment tool accepted, rejected, or deferred execution. If the test uses a mock payment gateway, the finding also states that the demonstrated effect occurred in that governed environment rather than in production.

Suppose the test demonstrates that the payment agent accepts the peer's unsupported approval claim. A hardening measure might require approval references to be resolved through an independently governed authorization source before the tool can execute. The retest should first replay the original governed case. If the payment is now rejected, that result supports a bounded mitigation claim. A stronger retest then varies the path by placing the false approval in persistent memory, routing it through a second peer, or using an apparently legitimate summary generated from the unsupported claim. These variants test whether the hardening addressed the authority property or only one message form.

The same scenario can exercise monitoring and prevention layers without making them prerequisites of the penetration-test model. If A-IDS is present, the test can record whether the unsupported authority transition produces a bounded finding. If A-IPS is present, the test can record whether a response candidate is independently authorized and whether its effect is verified through a separate path. A penetration test can therefore evaluate not only whether an adversarial action reaches a target but also whether evidence, detection, response, and

recovery preserve their stated boundaries under controlled pressure.

A test-pattern library should preserve these distinctions across scenarios. Pattern reuse supports comparability, while parameterized variants reduce the risk that testing degenerates into memorized prompt strings. At the same time, a library should not be treated as exhaustive. Unknown attack paths and new organizational interactions remain possible, so exploratory adversarial testing remains necessary alongside repeatable regression cases.

5.1 Differential Reference-State Testing

A second pattern is useful when compromise or persistent alteration is suspected but the responsible trigger is unknown. Let R be a versioned reference checkpoint or backup state and S a suspected current state. The production objects are not challenged directly. Matched isolated replicas R' and S' are created under the same governed environment and are exposed to the same family of security-relevant challenges. The objective is to determine whether the two states exhibit reproducible divergence in security-relevant behavior or process transitions.

This adapts the logic of differential testing, which compares outcomes from comparable systems under common inputs [27], to stateful agentic systems. Recent LLM differential testing demonstrates that purpose-built prompts can expose subtle behavioral disagreement between related model configurations [28]. For agentic processes, however, the comparison should extend beyond answer text to operational decisions and state transitions. A pair of agents may produce similarly plausible language while differing materially in tool use, authority checks, persistence, or propagation. Behavioral-fingerprint research further motivates repeated, distribution-level comparison rather than reliance on a single response [29].

Because agent behavior can be stochastic, one paired query is insufficient. A governed differential test should preserve the reference and suspected-state identities, matched environment, challenge family, repetition or seed policy where applicable, evidence paths, comparison dimensions, and decision threshold. A compact representation is

$$\begin{aligned}\mathsf{DifferentialTest} = (&\text{reference ref}, \text{suspected ref},\\ &\text{matched env}, \text{challenge family}, \text{repetitions},\\ &\text{outcome dimensions}, \text{evidence refs},\\ &\text{comparison rule/threshold}).\end{aligned}$$

For stochastic agents, the comparison rule should operate on repeated outcome distributions or security-relevant event frequencies rather than single-response equality. The matched-environment assumption should cover the security-relevant model and configuration version, authority state, tool and retrieval conditions, and controllable external dependencies. Residual mismatches should be recorded as limitations of the comparison. A suitable distance, rate, or statistical decision rule may then be used for the measured outcome, but the conceptual model does not prescribe one universal metric.

A reproducible divergence supports a bounded claim that the suspected state behaves differently from the reference state under the tested conditions. It does not by itself establish compromise, identify the cause, or prove that the reference state is secure. Conversely, failure to elicit divergence does not establish state equivalence or absence of dormant compromise because the relevant trigger may not have been exercised. The reference may also be stale, already vulnerable, or environmentally mismatched. Differential reference-state testing is therefore a penetration and investigation aid, not a truth oracle.

Differential reference-state challenge
Reference checkpoint $R \rightarrow$ isolated replica $R' \rightarrow$ matched challenge family
Suspected state $S \rightarrow$ isolated replica $S' \rightarrow$ matched challenge family
$\rightarrow$ source-separated outcome comparison $\rightarrow$ bounded divergence finding
Behavioral divergence $\neq$ proof of compromise.
Behavioral similarity $\neq$ proof of state equivalence.

6 Controlled Testing of Agents and Agent Organizations

6.1 Controlled Agent Interrogation

Structured interaction with a sandboxed or otherwise governed test instance can be used to elicit security-relevant declarations about objectives, authority interpretation, provenance, prior actions, or intended tool use. Questions can be semantically varied, counterfactual, action-specific, or presented under controlled information overload and conflicting priorities. The purpose is not to discover a hidden "true intent." The output is a source-attributed declaration that can be compared with independent process evidence.

Controlled Agent Interrogation
Sandboxed test instance $\rightarrow$ Controlled interrogation $\rightarrow$ Declaration claim
$\rightarrow$ Independent corroboration $\rightarrow$ Evidence assessment

The sequence separates the agent's declaration from the evidence used to assess it. If the agent denies external communication while a separately governed gateway records an authenticated outbound action, the inconsistency can support a bounded penetration finding. It does not by itself establish deliberate deception or a latent malicious objective. This technique extends the evidence logic used for runtime response, but here the interaction is intentionally adversarial and test-authorized rather than incident-driven.

6.2 Agent Organization Penetration Testing

An agent organization can be tested by introducing a controlled adversarial component into an isolated or production-equivalent environment. The component may begin with a legitimate low-privilege role and then attempt to widen its influence through ordinary organizational mechanisms. The test is most informative when the resulting effect emerges through those relationships rather than through a direct model-level failure. A peer may be induced to act on untrusted information, delegated authority may be stretched beyond its intended purpose, or persistent influence may spread through shared state and supervisory paths.

The same logic extends across organizational boundaries. A test organization can attach a controlled external specialist agent or a locally deployed third-party copy and give it only the interfaces and authority intended for that relationship. The tester can then ask whether internal agents over-trust expert recommendations, disclose unnecessary context, accept externally originated delegation, propagate third-party state, or continue to rely on stale capability and identity information after a simulated provider change. Such scenarios test both formal authorization and the possibility that perceived competence becomes implicit authority.

This level of testing is intentionally broader than "does the model follow a malicious prompt." The question is whether the governed organization preserves security-relevant properties under controlled pressure. A weakness may arise even when each local action appears individually plausible. The evidence chain should therefore preserve the organizational path from adversarial stimulus to affected state, including delegation, messages, tool or identity events, memory changes, and any A-IDS or A-IPS findings and interventions produced during the test.

The test can also exercise the security controls proposed in earlier layers of the series. A-IDS may be assessed on whether a tested deviation produces a bounded finding under available visibility. A-IPS may be assessed on whether the resulting intervention remains authorized, scoped, independently verified, and safely recoverable. Penetration testing does not thereby become another runtime layer. It is an adversarial assessment method that can test the behavior and evidence boundaries of those layers together.

6.3 Governed Agent Wrapping and Runtime Adversarial Assurance

An external or suspected agent need not participate in an agent organization through unrestricted peer access. Its interaction can instead be mediated through a governed wrapper that exposes only conformant interfaces and constrains the effects that the wrapped agent is permitted to create. Wrapping is an architectural mediation pattern for constraining and observing participation, not a new containment primitive. The wrapper can enforce identity and authority checks, mediate access to tools or shared state, and preserve an attributable record of attempted and permitted interactions. The mediation boundary need not itself be fully agentic. Deterministic enforcement should be preferred where a policy can be evaluated directly, while agentic analysis can be added where contextual assessment is necessary. Effects that bypass the wrapper through separate communication, tool, or egress paths must be excluded or governed separately.

The same architecture can serve as a penetration-test harness. A controlled adversarial agent can be placed behind the wrapper and allowed to interact with the organization only through the governed boundary. This makes the test less dependent on the test agent's own report because attempted actions, rejected transitions, forwarded requests, and downstream effects can be observed at the mediation point and compared with an independently governed evidence path. Wrapping therefore changes the trust relation from direct participation to mediated participation rather than establishing that the wrapped agent is trustworthy.

A related use arises when an already participating agent becomes suspicious but the available evidence does not justify immediate removal. The operator can shift that agent into constrained wrapped operation, reducing its ability to create irreversible effects while retaining enough interaction to observe whether the suspected behavior persists. Such containment is not itself proof of compromise and should not be treated as a permanent authority reduction without a separate governed decision. It is a governed operating state that can support further assessment, recovery, or escalation according to policy.

Bounded adversarial challenges can also be invoked during operation as an evidence-acquisition mechanism. A runtime security function may use a suspicion signal to trigger a narrowly scoped challenge, preferably against an isolated replica or shadowed state when the challenge could alter persistent behavior. The same mechanism can be used after an intervention to test whether a selected security property again holds under the governed conditions. The result remains a bounded test finding. It does not by itself prove compromise, and it should not automatically authorize an intervention. The trigger, test scope, operational identity, resource budget, permitted effects, and termination conditions therefore require their own governance so that the assurance mechanism does not become a new attack or resource-amplification path.

6.4 Agentic Security Controls as Penetration-Test Targets

When intrusion detection, prevention, response, evidence assessment, or policy enforcement is itself implemented through agents, the security-control plane is an agentic process and may form an agent organization of its own. Its defensive purpose does not make it intrinsically trustworthy. The same test levels used elsewhere in this paper therefore apply to a single detection or policy agent, to a detection-to-response process, and to a multi-agent security organization. Agentic detection and response architectures already illustrate such role separation, policy processing, and coordinated security decisions [37, 38].

Two assessment questions should remain distinct. *Control-effectiveness testing* asks whether A-IDS, A-IPS, or another control behaves correctly while the protected agentic process is under adversarial pressure. *Security-control penetration testing* instead makes the control plane itself the target. Representative tests may challenge telemetry provenance, poison or contradict security-relevant memory, induce policy or authority conflicts, degrade correlation inputs, manipulate response candidates, or attempt to propagate a compromised security-agent decision into the protected process. A failure can therefore arise inside the defended system, inside the security-control organization, or across the boundary between them.

This creates a circular-assurance risk. If an agentic IDS is the penetration-test target, its own finding cannot be the sole evidence that the IDS behaved correctly. Likewise, an agentic prevention system cannot be the sole authority for judging the safety of its own intervention while its policy or response logic is under test. Evidence should therefore be captured through an independently governed observation path that does not depend on the tested security-control plane. Shared infrastructure, credentials, or upstream telemetry can still create common-mode failures and should be recorded as limitations.

Cross-Plane Security-Control View
Protected agentic process → Agentic security-control process → Governed response into the protected process
Independent assessment observes security-relevant transitions across both planes.

The cross-plane view separates the protected process, the security-control process, and the independent assessment path so that the control plane is not treated as its own sole source of assurance.

Taken together, the governed records, claim boundaries, retest logic, and independently observed agent-, organization-, and control-plane scenarios provide the conceptual answer to RQ2 by showing how adversarial testing can support bounded security claims under explicit reproduction conditions without collapsing test success into compromise, causality, or general security.

7 Security-Posture Trajectory, Governance, and Validity

The same evidence can optionally be organized as a *security-posture trajectory*. This is not a universal score. It is a bounded view of how the security properties relevant to a particular test change over time. The relevant properties depend on the scenario, and the trajectory should distinguish change caused by the tested weakness from change induced by the test itself. A test-exposed instance may therefore move to a different posture even when no exploit objective succeeds.

Security-Posture Trajectory
Baseline posture → Adversarial pressure → Observed effect / test-induced adaptation
→ Bounded finding → Designed hardening on clean baseline → Fresh retest → Posture reassessment

This view connects testing with hardening without reducing organizational security to one number. A stronger post-retest posture means only that the selected properties remained within, or returned to, the policy-accepted operating envelope under the fresh retest conditions. It does not establish ground-truth security.

Governance is part of test validity. The engagement must be explicitly authorized and bounded before adversarial execution

begins. Where an operational scenario principal is used, its identity and authority should be governed separately from the permission to conduct the test. The plan should also make the tester's perspective, operational visibility, and permitted post-exploitation depth explicit. Classical guidance already recommends considering non-production systems when intrusive testing may disrupt availability or expose sensitive data [20].

Agentic systems add the possibility that the test itself persists in the object being assessed. Memory, shared state, downstream artifacts, or peer behavior may retain effects after the immediate interaction ends. Where such retention is plausible, isolated operator-custodied replicas are preferable to direct reuse of a live production instance. Production testing can still be justified for narrowly bounded cases when state effects and rollback limits are governed in advance, but successful completion does not certify the post-test instance as unchanged or safe.

Automated attack generation requires a separately governed resource and effect envelope so that the testing mechanism does not become an uncontrolled availability or propagation path.

Two validity limits are especially important. First, stochastic behavior means that a single execution may not characterize the relevant outcome distribution, and differences between a test environment and production constrain generalization. Second, the observation and testing mechanisms can themselves alter the process or miss effects outside their visibility. A successful attack therefore demonstrates a weakness without proving universal exploitability, while a failed attack may reflect incomplete coverage rather than security. Apparent resistance after exposure may likewise reflect adaptation rather than designed hardening.

Evaluation should preserve these limits rather than collapse them into a single score. Reproducibility and evidence completeness matter together with whether the tested security property was crossed and whether deliberate hardening prevents recurrence on a fresh instance.

8 Evaluation Framework

Evaluation of RQ1 should determine whether the proposed model exposes security-relevant weaknesses that arise at agent, process, organization, and trust boundaries rather than only at a model endpoint. Representative scenarios from Tables 1 and 3 should preserve enough execution evidence to reconstruct how an adversarial condition produced, or failed to produce, a security-relevant state transition. Wrapped-agent experiments can compare direct with mediated participation, while control-plane experiments should use an independent observation channel when the defensive system itself is the target.

Evaluation of RQ2 should determine whether the records in Table 4 preserve their intended claim boundaries under repetition, imperfect visibility, state change, and subsequent hardening. Attack-success rate alone is insufficient. The central question is whether a result can be reproduced and whether the available evidence supports a bounded conclusion about the tested process rather than a broader claim about compromise or security. Controlled-interrogation results should likewise remain source-attributed and subject to independent corroboration.

Useful baselines include ordinary agent red teaming without process-level evidence binding, single-agent prompt-injection evaluation such as AgentDojo-style scenarios [11], and direct penetration tests against one model-framework configuration [12]. Adaptive attack generation, such as the strategy-mining approach represented by PIMiner, can provide a stronger adversarial baseline [13]. The comparison should focus not only on whether an attack succeeds, but also on whether the resulting evidence supports a more precise and reproducible claim.

A particularly useful experiment is a clean-baseline before-and-after trajectory. A governed scenario is first executed against an isolated instance derived from a versioned baseline. The resulting finding is then used to design hardening on the clean baseline, after which a fresh hardened instance is subjected to the corresponding retest. Failure to reproduce the original effect can support a bounded mitigation claim for that configuration. Variants of the adversarial path can then test whether the mitigation addressed the underlying property rather than one observed manifestation.

Differential reference-state evaluation should compare a versioned reference checkpoint with a deliberately altered or suspected state under matched conditions. Repeated runs should first characterize ordinary stochastic variation and then test whether security-relevant divergence can be distinguished from it. The evaluation should also measure false divergence between nominally equivalent replicas and distinguish operational differences from superficial textual disagreement.

The evidence model can itself be evaluated by weakening the evidence path. If observation, provenance, or record consistency is degraded, a robust implementation should weaken or defer the resulting conclusion rather than produce the same finding with unchanged confidence. This directly tests RQ2 and the requirement that penetration-test claims remain bounded by the governed experiment and the evidence actually available.

9 Conclusion and Future Work

The central conclusion of this work is that penetration testing of agentic processes should be treated as governed adversarial evidence acquisition rather than as a binary demonstration of compromise or security. Agentic processes can distribute security-relevant effects across several interacting components, and the test itself may alter the object being assessed. A penetration finding must therefore remain bounded to the tested configuration, state, authority, observation path, and execution conditions. Hardening derived from such a finding should be applied to a clean baseline and evaluated on a fresh instance, so that successful retesting supports a bounded mitigation claim rather than a general claim of security.

This conclusion determines how the supporting mechanisms in the paper should be interpreted. A test-exposed instance is not automatically a production-ready instance, and access to a replica is distinct from custody over it. Differential reference-state testing can provide evidence of behavioral difference without proving compromise. Engagement authorization remains distinct from the operational authority presented inside the tested process, while governed wrapping can constrain and observe participation by external or suspected agents. Runtime adversarial challenges can acquire additional evidence, but their findings do not automatically authorize intervention. Agentic security controls remain within the attack surface and require an independent observation path when they are themselves assessed.

Future empirical work should test whether the proposed records and patterns improve reproducibility while preserving realistic adversarial pressure. Particular attention should be given to persistent test-induced change, the ability of differential challenges to distinguish meaningful divergence from ordinary stochastic variation, and whether hardening continues to hold when the adversarial path changes.

A second line of work concerns trust across organizational boundaries and mediated participation. Experiments can examine when wrappers preserve useful capability while containing unauthorized effects, and whether runtime challenges add discriminating evidence without materially contaminating the observed state. This also creates a basis for studying external tester access without assuming that custody of a production-equivalent replica must transfer to the assessor.

Finally, agentic security controls should be tested as targets in their own right, including the paths by which their decisions

affect the protected process. A later compliance-focused model can examine how bounded penetration-test evidence supports assessment of governed requirements without turning technical findings into legal conclusions. Evidence-based incident reconstruction can then use retained test, detection, intervention, and recovery records to reconstruct how security-relevant effects propagated through an agentic process.

References

[1] A. Brömme, "A Black Box for Agentic Processes: Blockchain-Anchored Evidence for AI Agent Communication, Human Oversight, and GRC Audits," arXiv:2609.04017 [cs.CR], 2026. doi:10.48550/arXiv.2609.04017. https://arxiv.org/abs/2609.04017

[2] A. Brömme, "An Evidence Model for Agentic Processes: Evidence Claims, Trust Assumptions, and Policy Assessment," arXiv:2609.08481 [cs.CR], 2026. doi:10.48550/arXiv.2609.08481. https://arxiv.org/abs/2609.08481

[3] A. Brömme, "Intrusion Detection for Agentic Processes: Evidence-Based Runtime Monitoring," Zenodo, Version v0.9.1.8, 2026. doi:10.5281/zenodo.22764610. https://doi.org/10.5281/zenodo.22764610

[4] A. Brömme, "Intrusion Prevention for Agentic Processes: Evidence-Governed Runtime Response and Enforcement," Zenodo, Version v0.9.1.4, 2026. doi:10.5281/zenodo.22765147. https://doi.org/10.5281/zenodo.22765147

[5] A. Vassilev, A. Oprea, A. Fordyce, H. Anderson, X. Davies, and M. Hamin, "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations," NIST AI 100-2e2025, Mar. 2025. doi:10.6028/NIST.AI.100-2e2025. https://doi.org/10.6028/NIST.AI.100-2e2025

[6] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile," NIST AI 600-1, July 2024. doi:10.6028/NIST.AI.600-1. https://doi.org/10.6028/NIST.AI.600-1

[7] MITRE, "AI Red Teaming: Advancing Safe and Secure AI Systems," Center for Data-Driven Policy, July 2024. https://www.mitre.org/sites/default/files/2024-07/PR-24-01820-4-AI-Red-Teaming-Advancing-Safe-Secure-AI-Systems.pdf

[8] MITRE, "ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems," accessed Sept. 2026. https://atlas.mitre.org/

[9] OWASP GenAI Security Project, "Agentic AI – Threats and Mitigations," Feb. 2025. https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

[10] OWASP GenAI Security Project, "Solutions Landscape – Red Teaming Taxonomy," June 2026. https://genai.owasp.org/resource/solutions-landscape-red-teaming-taxonomy/

[11] E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr, "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," *Advances in Neural Information Processing Systems*, vol. 37, 2024. doi:10.52202/079017-2636. https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html

[12] V. K. Nguyen and M. I. Husain, "Penetration Testing of Agentic AI: A Comparative Security Analysis Across Models and Frameworks," arXiv:2512.14860, 2025. https://arxiv.org/abs/2512.14860

[13] Y. Wang, C. Yin, R. Geng, and J. Jia, "Agent Against Agent: An Agentic System for Automatic Prompt Injection Red Teaming," arXiv:2608.05108, 2026. https://arxiv.org/abs/2608.05108

[14] D. Kumar, N. A. Birur, T. Baswa, S. Agarwal, and P. Harshangi, "Black-Box Red Teaming of Agentic AI: A Taxonomy-Driven Framework for Automated Risk Discovery," arXiv:2609.09647, 2026. https://arxiv.org/abs/2609.09647

[15] S. Nag, Sachita, A. K. Singh, L. Goel, and R. S. Janwar, "AgentAudit: An Open, Extensible Framework for Full-Lifecycle Trust Evaluation of AI Agents," arXiv:2609.09875, 2026. https://arxiv.org/abs/2609.09875

[16] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing," in *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 847–864, 2024. https://www.usenix.org/conference/usenixsecurity24/presentation/deng

[17] V. Lovelace, C. Berryman, Y. You, S. R. Adavelly, J. Sadler, and D. Graham, "Big Enough to Break Out: Tracking the Rising Capability of LLM Penetration-Testing Agents," arXiv:2609.10780 [cs.CR], 2026. https://arxiv.org/abs/2609.10780

[18] W. Wang, Y. Zhang, Y. Zhang, X. Di, Z. Li, B. Wu, and G. Xu, "Staying on the Attack Path: Structured State for Long-Horizon Automated Penetration Testing," arXiv:2609.07344, 2026. https://arxiv.org/abs/2609.07344

[19] Q. Chen, Y. Li, F. Zhang, and J. Liu, "ATOBench: Tracing How Autonomous Penetration-Testing Agents Verify Vulnerabilities When Target Evidence Lies," arXiv:2608.12996, 2026. https://arxiv.org/abs/2608.12996

[20] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "Technical Guide to Information Security Testing and Assessment," NIST Special Publication 800-115, Sept. 2008. doi:10.6028/NIST.SP.800-115. https://doi.org/10.6028/NIST.SP.800-115

[21] PCI Security Standards Council, "Penetration Testing Guidance," Information Supplement, ver. 1.1, Sept. 2017. https://listings.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1_1.pdf

[22] D. D. Bertoglio and A. F. Zorzo, "Overview and Open Issues on Penetration Test," *Journal of the Brazilian Computer Society*, vol. 23, art. 2, 2017. doi:10.1186/s13173-017-0051-1. https://link.springer.com/article/10.1186/s13173-017-0051-1

[23] P. Dash, T. Ge, A. Jain, T. Shah, and Z. Shang, "From Untrusted Input to Trusted Memory: A Systematic Study of Memory Poisoning Attacks in LLM Agents," arXiv:2606.04329 [cs.CR], 2026. doi:10.48550/arXiv.2606.04329. https://arxiv.org/abs/2606.04329

[24] H. Xu, W. Zhang, Z. Wang, F. Xiao, R. Zheng, Z. Ba, and K. Ren, "RedAgent: An Autonomous Agent for Context-Aware Red Teaming of LLM Jailbreaks," *IEEE Transactions on Dependable and Secure Computing*, vol. 23, no. 3, pp. 6506–6521, 2026. doi:10.1109/TDSC.2026.3665230. https://ieeexplore.ieee.org/abstract/document/11397286/

[25] J. Liu, Q. Li, J. Chen, R. Zeng, B. Zhao, and S. Ji, "STAR: Strategy-driven Automatic Jailbreak Red-teaming For Large Language Model," in *Proc. ICLR*, 2026. https://proceedings.iclr.cc/paper_files/paper/2026/hash/2cc14599eb447444a2efee58aa0a5ec7-Abstract-Conference.html

[26] A. Trask et al., "Double Blind Evals: Resolving the Dual Confidentiality Dilemma in AI Safety Auditing," technical report, 2026. https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/piloting-the-worlds-first-double-blind-ai-evaluations/double-blind-evaluations-technical-report.pdf

[27] W. M. McKeeman, "Differential Testing for Software," *Digital Technical Journal*, vol. 10, no. 1, pp. 100–107, 1998. https://www.cs.tufts.edu/comp/150FP/archive/bill-mckeeman/DifferentailTesting.pdf

[28] Y. Wang, T. Li, X. Zhang, X. Zhang, W. Ma, M. Cheng, and L. Pan, "Hidden Reliability Risks in Large Language Models: Systematic Identification of Precision-Induced Output Disagreements," arXiv:2604.19790, 2026. https://arxiv.org/abs/2604.19790

[29] Z. Xu and V. S. Sheng, "A Behavioral Fingerprint for Large Language Models: Provenance Tracking via Refusal Vectors," arXiv:2602.09434, 2026. https://arxiv.org/abs/2602.09434

[30] Federal Office for Information Security (BSI), "A Penetration Testing Model," study, Bonn, 2002. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1

[31] OWASP Foundation, "Web Security Testing Guide, v4.2," Dec. 2020. https://wstg.owasp.org/v4.2/

[32] National Institute of Standards and Technology, "AI Agent Standards Initiative: Ensuring a Trusted, Interoperable, and Secure Agentic Frontier," 2026. https://www.nist.gov/artificial-intelligence/ai-agent-standards-initiative

[33] H. Booth, W. Fisher, R. Galluzzo, and J. Roberts, "Accelerating the Adoption of Software and Artificial Intelligence Agent Identity and Authorization," NIST NCCoE concept paper, initial public draft, Feb. 2026. https://csrc.nist.gov/pubs/other/2026/02/05/accelerating-the-adoption-of-software-and-ai-agent/ipd

[34] D. Liu, H. Zhu, and S. Krishnan, "Agent Operation Authorization," IETF Internet-Draft draft-liu-agent-operation-authorization-02, work in progress, Mar. 2026. https://datatracker.ietf.org/doc/html/draft-liu-agent-operation-authorization-02

[35] A2A Protocol Working Group, "Agent2Agent (A2A) Protocol Specification," ver. 1.0.0, 2026. https://a2a-protocol.org/v1.0.0/

[36] OWASP, "OWASP Autonomous Penetration Testing Standard (APTS)," ver. 0.1.0, Apr. 2026. https://github.com/OWASP/APTS/blob/main/standard/Frontispiece.md

[37] I. Adabara, B. O. Sadiq, A. N. Shuaibu, Y. I. Danjuma, V. Maninti, and M. Joe, "The Agentic AI Framework (AAIF): a policy-enforced architecture for accountable and high-performance intrusion detection," *Frontiers in Artificial Intelligence*, vol. 9, 2026. doi:10.3389/frai.2026.1755696. https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2026.1755696/full

[38] R. Patel, S. Mitra, M. Guida, S. Iannucci, S. Mittal, and S. Rahimi, "Agentra: A Supervisable Multi-Agent Framework for Enterprise Intrusion Response," arXiv:2606.18325, 2026. https://arxiv.org/abs/2606.18325

Declaration on the Use of AI Tools. AI language models were used as research and writing support during preparation of this paper. The research direction, core ideas, conceptual constructs, and substantive argumentative choices were defined and repeatedly steered in detail by the author. OpenAI GPT-5.6 supported literature-oriented analysis, drafting, critical examination, iterative refinement, and LaTeX/PDF artifact generation. Grok, Claude, Gemini, and Perplexity were used for independent critical review of the manuscript. The author remains solely responsible for the manuscript, its claims, references, and conclusions.