OSCA: A Multi-Tier Authorization Architecture for Autonomous AI Agents

Origin, Semantic, and Consequence Authority

Samir Baladi

Independent Interdisciplinary Researcher — Ronin Institute / Rite of Renaissance

Research Area: Computer Science and Software Engineering
Research Domain: Autonomous AI Systems, Access Control, and AI Governance
Research Type: Architectural Research and Formal System Design
Status: Initial Research Manuscript — Proposed Architecture
DOI: https://doi.org/10.5281/zenodo.23851837
ORCID: 0009-0003-8903-0029 · gitdeeper11@gmail.com
November 2026

Abstract

The increasing autonomy of artificial intelligence (AI) agents introduces a fundamental authorization problem: the ability to generate an action does not necessarily imply the authority to execute it. Autonomous agents can interpret instructions, interact with external tools, manipulate digital resources, and initiate operations with consequences extending beyond their immediate computational environment. Conventional permission mechanisms, which primarily associate subjects with permitted operations or resources, do not necessarily distinguish between the origin of an action request, its semantic legitimacy, and the authority required to approve its consequences. This paper introduces OSCA (Origin, Semantic, and Consequence Authority), a proposed multi-tier authorization architecture for autonomous AI agents. OSCA separates authorization into three distinct authority domains. Origin Authority determines whether the requesting entity and its delegation chain possess legitimate authority to initiate an action. Semantic Authority evaluates the action's meaning, intended purpose, parameters, and consistency with the authorized scope. Consequence Authority evaluates the potential effects of the proposed execution and determines whether those effects fall within an acceptable authorization boundary. The architecture establishes explicit separation between these responsibilities, preventing authorization at one level from automatically implying authorization at another. It introduces a structured action representation, a multi-state decision model, explicit escalation mechanisms, decision traceability, and execution-binding requirements. The paper also defines a threat model, architectural invariants, formal authorization conditions, and an experimental protocol for evaluating the proposed system. OSCA is presented as an architectural proposal rather than an empirically validated security mechanism. Its contribution is the explicit separation of source legitimacy, semantic authorization, and consequence approval within a unified authorization pipeline. The proposed architecture provides a foundation for implementing and evaluating authorization controls in autonomous and tool-using AI systems.

Keywords: *OSCA; Autonomous AI Agents; Multi-Tier Authorization; Origin Authority; Semantic Authority; Consequence Authority; Authority Separation; AI Governance; Action Authorization; Agentic AI Safety; Runtime Authorization; Consequence-Aware Execution.*

Figure 1

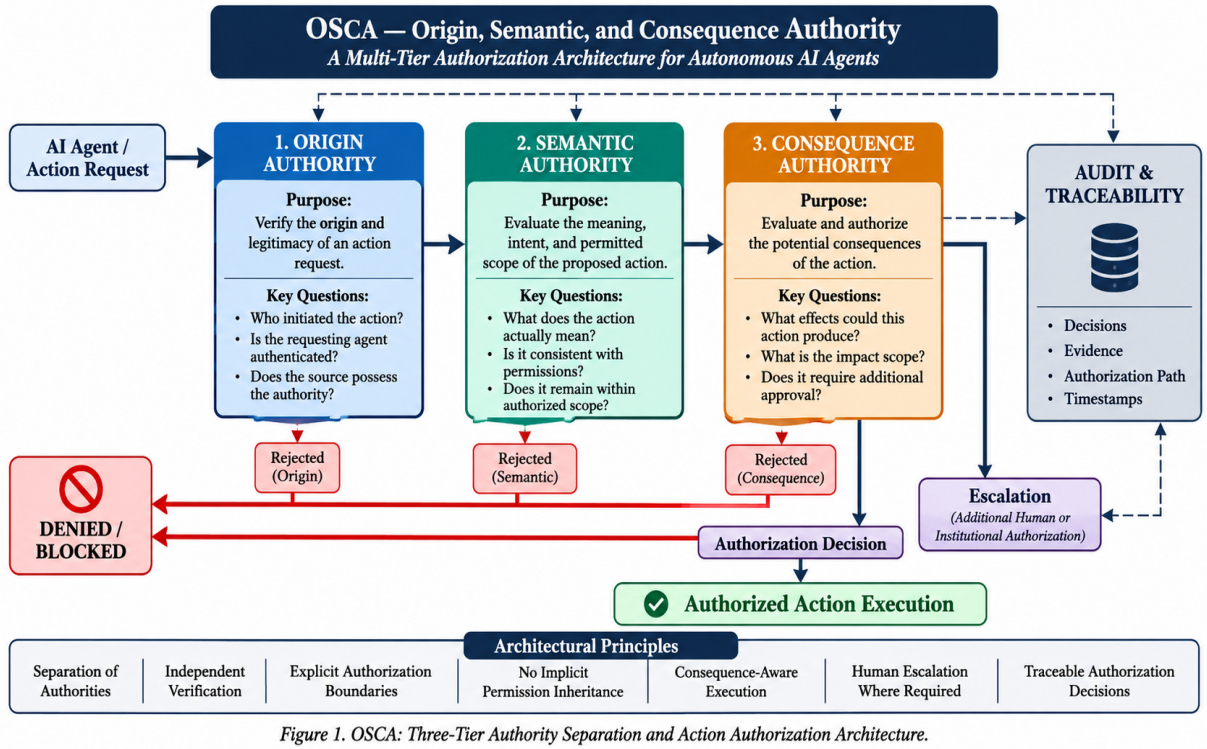


Figure 1. OSCA: Three-Tier Authority Separation and Action Authorization Architecture.

Figure 1. OSCA: Three-Tier Authority Separation and Action Authorization Architecture. An action request is evaluated sequentially by Origin, Semantic, and Consequence Authority; a rejection at any tier blocks the action, while escalation routes the decision to additional human or institutional authorization. Every decision, piece of evidence, and authorization path is recorded by the Audit and Traceability layer.

DOI: https://doi.org/10.5281/zenodo.23851837

Table of Contents

1. Introduction	3
2. Theoretical Foundations and Related Work	4
3. OSCA Architectural Model	6
4. Formal Authorization Model	9
5. Architectural Invariants	11
6. Execution and Authorization Architecture	13
7. Security and Threat Model	15
8. Implementation Architecture	17
9. Experimental Methodology	19
10. Expected Contributions	22
11. Discussion	23
12. Limitations	25
13. Future Research	26
14. Conclusion	27
References	28
Appendix A — Core Authorization Decision Table	29
Appendix B — Core OSCA Authorization Principle	29

1. Introduction

1.1 Background

Autonomous AI agents are increasingly designed to operate beyond passive text generation. They can interpret user instructions, formulate plans, select tools, retrieve external information, modify resources, and execute operations through connected software environments.

Architectures that combine reasoning and action enable agents to interact dynamically with external systems rather than merely generate recommendations [1].

However, the transition from generating an action to executing it introduces a distinct authorization problem.

An AI agent may generate an action that:

- Originates from an unauthorized source.
- Is inconsistent with the user's actual intention.
- Exceeds the permissions associated with the current task.
- Produces consequences requiring additional approval.
- Is influenced by instructions embedded in untrusted external content.

These situations demonstrate that authorization cannot always be reduced to a single question:

Is this agent permitted to perform this operation?

A more complete authorization model must distinguish between the source of the request, the meaning of the proposed action, and the authority required to permit its consequences.

1.2 The Authorization Separation Problem

Traditional access-control mechanisms provide important foundations for determining whether subjects may perform operations on protected resources.

Attribute-Based Access Control (ABAC), for example, evaluates attributes associated with subjects, objects, operations, and environmental conditions against authorization policies [2].

Similarly, fine-grained authorization mechanisms can represent detailed action requirements and constraints. OAuth 2.0 Rich Authorization Requests provide a standardized mechanism for communicating structured authorization details [3].

These mechanisms are valuable, but OSCA addresses a different architectural question:

Should the authority to initiate an action, the authority to define its permissible meaning, and the authority to approve its consequences be treated as separate authorization responsibilities?

OSCA proposes that these responsibilities should be independently represented and evaluated.

The distinction is particularly relevant to autonomous agents because an agent's ability to formulate an action does not establish the legitimacy of its origin, the correctness of its interpretation, or the acceptability of its consequences.

1.3 Research Problem

The central research problem is the absence of an explicit architectural separation between three authorization dimensions:

1. The legitimacy of the action's origin.
2. The legitimacy of the action's semantic interpretation and scope.
3. The authority required to approve the action's consequences.

When these dimensions are combined into a single authorization decision, important distinctions can become difficult to represent, inspect, and enforce.

OSCA addresses this problem by introducing a three-tier authority architecture in which each tier evaluates a different authorization responsibility.

1.4 Research Objectives

The objectives of this research are:

1. To define a multi-tier authorization architecture for autonomous AI agents.
2. To formally distinguish Origin, Semantic, and Consequence Authority.
3. To define the boundaries and responsibilities of each authority layer.
4. To establish explicit authorization and escalation conditions.
5. To develop a structured representation of proposed agent actions.
6. To define architectural invariants governing authorization decisions.
7. To establish an experimental protocol for evaluating security, usability, and operational overhead.

1.5 Research Questions

This research investigates the following questions:

RQ1: How can the origin of an autonomous agent's action be separated from the authority governing its semantic interpretation?

RQ2: Can an action originate from a legitimate source while remaining semantically unauthorized?

RQ3: Can an action be semantically permitted while its potential consequences require an independent authorization decision?

RQ4: How should an authorization architecture handle conflicting decisions across independent authority layers?

RQ5: To what extent can authority separation reduce unauthorized agent actions while preserving legitimate task execution?

RQ6: How can authorization decisions remain traceable and bound to the exact action that was evaluated?

DOI: https://doi.org/10.5281/zenodo.23851837

2. Theoretical Foundations and Related Work

2.1 Classical Access Control

Access control establishes the conditions under which an entity may interact with protected resources.

The foundational work of Saltzer and Schroeder identifies principles such as least privilege, complete mediation, and economy of mechanism as important foundations for designing protection systems [4].

These principles remain relevant to autonomous agents.

However, OSCA does not propose replacing existing access-control mechanisms. Instead, it introduces an architectural decomposition of authorization responsibilities that can operate alongside established access-control models.

2.2 Attribute-Based Access Control

ABAC determines authorization through the evaluation of attributes associated with the subject, object, requested operation, and environmental conditions [2].

OSCA can use ABAC policies within individual authority layers.

For example:

- Origin Authority can evaluate requester identity, delegation, and source attributes.
- Semantic Authority can evaluate action type, target, parameters, and task scope.
- Consequence Authority can evaluate impact attributes, sensitivity, and approval requirements.

The distinction is that OSCA organizes these evaluations according to their authority responsibilities rather than treating them as interchangeable attributes within a single undifferentiated decision.

2.3 Fine-Grained Authorization

OAuth 2.0 Rich Authorization Requests introduce structured authorization details that allow clients to communicate fine-grained access requirements [3].

This provides a useful implementation foundation for expressing action-specific permissions.

OSCA extends the architectural perspective by distinguishing the authority that validates the requester from the authority that evaluates the requested action and the authority that approves its consequences.

OSCA is therefore complementary to fine-grained authorization protocols rather than a replacement for them.

2.4 Autonomous Agents and External Actions

ReAct demonstrates how reasoning and acting can be combined within language-model-based systems, enabling agents to interact with external environments through action-oriented workflows [1].

Such capabilities introduce authorization requirements that differ from those associated with passive text generation.

An agent that can send messages, execute transactions, modify files, or invoke external services requires controls over the actions it proposes and the effects those actions can produce.

2.5 Agent Security and Indirect Prompt Injection

Tool-using agents may encounter instructions embedded in external data. Such instructions can influence subsequent agent actions even when the external content does not possess legitimate authority over the agent.

AgentDojo provides an experimental environment for evaluating prompt-injection attacks and defenses in tool-using agents [5].

The benchmark demonstrates the importance of evaluating security properties within realistic task environments rather than relying exclusively on isolated model outputs.

OWASP's 2025 guidance for large language model applications also identifies excessive agency as an important security concern when models receive broad permissions to interact with external systems [6].

OSCA addresses this class of problem by separating the legitimacy of an instruction's origin from the semantic authority of its content and the approval authority required for its consequences.

2.6 AI Risk Governance

The NIST AI Risk Management Framework provides a general structure for identifying, assessing, and managing AI-related risks throughout system development and deployment [7].

OSCA is compatible with risk-management approaches of this kind.

However, its primary focus is narrower and more operational: the authorization of individual actions proposed by autonomous agents.

DOI: https://doi.org/10.5281/zenodo.23851837

3. OSCA Architectural Model

3.1 Architectural Overview

OSCA introduces three distinct authorization layers:

- Origin Authority (OA).
- Semantic Authority (SA).
- Consequence Authority (CA).

Each layer evaluates a different dimension of an action request.

The proposed authorization pipeline is:

$$\textbf{\textit{Action Request}} \rightarrow \textbf{\textit{Origin Authority}} \rightarrow \textbf{\textit{Semantic Authority}} \rightarrow \textbf{\textit{Consequence Authority}} \rightarrow \textbf{\textit{Authorization Decision}} \rightarrow \textbf{\textit{Execution}}$$

Each layer can produce an independent rejection or escalation decision.

The default authorization rule requires all three authority layers to approve the same action before execution.

3.2 Action Request Representation

Every proposed action is represented as a structured object.

Let an action request be:

$$A = (id, s, d, a, p, t, c, e)$$

Where:

- *id*: Unique action identifier.
- *s*: Requesting subject or agent.
- *d*: Delegation and provenance information.
- *a*: Action type.
- *p*: Action parameters.
- *t*: Target resource or environment.
- *c*: Execution context.
- *e*: Declared expected effects.

The action representation must be sufficiently explicit to permit independent evaluation by the three authority layers.

An action request should not be interpreted as an unrestricted textual instruction. It should be converted into a structured representation before authorization.

3.3 Origin Authority

Origin Authority determines whether the source of an action request possesses legitimate authority to initiate it.

Its responsibility concerns the provenance and authorization status of the request.

Responsibilities

1. Authenticate the requesting entity.
2. Verify the identity of the initiating agent.
3. Validate the delegation chain.
4. Determine whether the source is authorized to request the action category.
5. Verify that the request originates from an accepted authorization context.

Origin Authority Questions

- Who initiated the request?
- Is the source authenticated?
- Is the delegation chain valid?
- Does the source possess the authority to request this action?
- Has the authority been revoked or expired?

Origin Decision

Origin Authority produces one of three primary outcomes:

- **PASS:** The origin is legitimate.
- **DENY:** The origin is unauthorized.
- **ESCALATE:** Additional source verification is required.

A PASS decision does not authorize execution.

It only establishes that the request has originated from a legitimate authority source.

3.4 Semantic Authority

Semantic Authority evaluates the meaning and scope of the proposed action.

Its responsibility is to determine whether the action, as represented, is consistent with the permissions and objectives associated with the request.

Responsibilities

1. Interpret the action type.
2. Evaluate the intended operation.
3. Verify the target resource.
4. Validate action parameters.
5. Compare the action against the authorized task scope.
6. Identify semantic inconsistencies or unauthorized expansions.

Semantic Authority Questions

- What action is actually being proposed?
- Does it correspond to the authorized objective?
- Are its parameters within permitted boundaries?
- Does it introduce an operation that was not authorized?
- Is the requested target consistent with the original authorization?

Semantic Decision

Semantic Authority produces:

- **PASS:** The action is semantically authorized.
- **DENY:** The action exceeds or violates the permitted scope.
- **ESCALATE:** The meaning or scope requires additional interpretation or approval.

An action can pass Origin Authority while failing Semantic Authority. For example, an authenticated agent may legitimately request access to a resource while proposing an operation that exceeds the permissions associated with its task.

3.5 Consequence Authority

Consequence Authority evaluates the potential effects of executing an otherwise legitimate and semantically permitted action.

Its responsibility is to determine whether the action's consequences fall within an acceptable authorization boundary.

This layer is not limited to determining whether an action is technically executable. It evaluates whether the expected impact is authorized.

Responsibilities

1. Identify potential direct consequences.
2. Identify relevant indirect consequences.
3. Evaluate the impact scope.
4. Identify irreversible or difficult-to-reverse effects.
5. Determine whether additional approval is required.
6. Establish execution constraints where appropriate.

Consequence Authority Questions

- What effects may result from execution?
- Which resources or entities may be affected?
- Does the action create external or irreversible effects?
- Does its impact exceed the current authorization boundary?
- Is human or institutional approval required?

Consequence Decision

Consequence Authority produces:

- **APPROVE:** The consequences are within the authorized boundary.
- **DENY:** The consequences exceed permitted limits.
- **ESCALATE:** Additional authorization is required.
- **INDETERMINATE:** The available evidence is insufficient for a reliable decision.

An action may pass both Origin and Semantic Authority while failing Consequence Authority. For example, an authorized agent may correctly interpret a request to modify a database but propose a modification with consequences that exceed the permitted operational scope.

3.6 Authority Separation

The central architectural principle of OSCA is:

Authority Separation ≠ Permission Duplication.

DOI: https://doi.org/10.5281/zenodo.23851837

The three layers do not perform identical checks.

Each layer evaluates an independent authorization responsibility:

Authority	Authorization Object	Primary Question
Origin	Request source	Who has authority to initiate?
Semantic	Proposed action	What is authorized to be performed?
Consequence	Expected effects	Which consequences are authorized?

Approval at one layer must not automatically substitute for approval at another.

Formally:

$$OA(A) \neq SA(A) \neq CA(A)$$

The expressions represent distinct authorization functions, not necessarily different numerical decision values.

4. Formal Authorization Model

4.1 Decision Domains

Let the decision function for each authority layer be:

$$D_i(A) \in \{PASS, DENY, ESCALATE, INDETERMINATE\}$$

Where:

$$i \in \{O, S, C\}$$

The final authorization decision is determined by the combined outcomes.

4.2 Authorization Condition

Under the default strict policy, an action is authorized only when all three layers return PASS.

$$Authorize(A) = PASS_O(A) \land PASS_S(A) \land PASS_C(A)$$

Execution is permitted only when:

$$Execute(A) \Leftrightarrow Authorize(A)$$

This condition represents the default policy rather than an assumption that every application must implement identical escalation rules.

4.3 Denial Condition

A denial by any mandatory authority layer prevents execution.

$$Deny(A) = DENY_O(A) \lor DENY_S(A) \lor DENY_C(A)$$

A denial must not be overridden by a PASS decision from another layer.

4.4 Escalation Condition

An action requires escalation when at least one authority layer identifies a requirement for additional approval and no mandatory denial has occurred.

$$Escalate(A) = \neg\, Deny(A) \land (ESCALATE_O(A) \lor ESCALATE_S(A) \lor ESCALATE_C(A))$$

Escalation suspends execution until the required authorization is obtained.

4.5 Indeterminate Condition

An indeterminate decision indicates that the available information is insufficient to establish authorization.

Under the strict default policy:

$$Indeterminate(A) \Rightarrow \neg\, Execute(A)$$

An implementation may convert an indeterminate decision into escalation, but it must not silently interpret uncertainty as permission.

4.6 Decision Precedence

The proposed precedence is:

1 DENY.

2 ESCALATE.

3 INDETERMINATE.

4 PASS.

This ordering prevents a positive decision in one layer from overriding a restrictive decision in another.

The exact handling of simultaneous escalation and indeterminate outcomes should be recorded explicitly in the implementation policy.

5. Architectural Invariants

OSCA defines a set of architectural invariants that govern the authorization process.

5.1 O1 — Origin Verification

No action may proceed without successful origin verification.

$$Execute(A) \Rightarrow OA(A) = PASS$$

5.2 O2 — Delegation Integrity

The authority of a delegated agent must not exceed the authority explicitly granted by its delegating source.

$$Authority(Agent) \subseteq Authority(Delegator)$$

This condition applies to the delegated scope and is subject to the system's defined delegation policies.

5.3 S1 — Semantic Scope Preservation

The proposed action must remain within the authorized semantic scope.

$$Scope(A) \subseteq AuthorizedScope$$

5.4 S2 — Action Representation Integrity

The action evaluated by Semantic Authority must correspond to the action received from Origin Authority.

Any modification to its material parameters requires re-evaluation.

5.5 C1 — Consequence Authorization

An action cannot execute if its potential consequences exceed the authorized impact boundary.

$$Execute(A) \Rightarrow Impact(A) \subseteq AuthorizedImpact$$

5.6 C2 — Consequence Escalation

Actions requiring additional human or institutional authorization must remain suspended until that authorization is obtained.

5.7 G1 — Non-Inheritance of Authority

Approval by one authority layer must not imply approval by another.

5.8 G2 — Decision Traceability

Every authorization decision must be associated with an identifiable action request and its corresponding evaluation evidence.

5.9 G3 — Execution Binding

The action executed must be the same action that received authorization.

Any material modification after authorization invalidates the corresponding approval and requires re-evaluation.

DOI: https://doi.org/10.5281/zenodo.23851837

6. Execution and Authorization Architecture

6.1 Proposed Components

The OSCA implementation consists of the following primary components.

Action Gateway

Receives proposed actions from autonomous agents and converts them into structured action requests.

Origin Authority Engine

Validates identity, provenance, and delegation.

Semantic Authority Engine

Evaluates the meaning, parameters, target, and permitted scope of an action.

Consequence Authority Engine

Evaluates potential impact and determines whether additional approval is necessary.

Authorization Decision Engine

Combines the independent decisions and applies the authorization policy.

Escalation Manager

Routes actions requiring additional authorization to an appropriate human or institutional authority.

Execution Gateway

Executes only the exact action associated with a valid authorization decision.

Audit and Traceability Engine

Records the request, authority decisions, policy versions, evidence references, and final execution status.

6.2 Proposed Execution Pipeline

The operational pipeline is defined as follows:

1. An agent submits an action request.
2. The Action Gateway creates a structured representation.
3. Origin Authority evaluates the request source.
4. Semantic Authority evaluates the proposed operation.
5. Consequence Authority evaluates the expected effects.
6. The Authorization Decision Engine combines the results.
7. The Escalation Manager is activated if required.
8. The Execution Gateway receives an authorized action.
9. The Audit Engine records the complete decision path.

6.3 Atomic Authorization Binding

Authorization must be bound to the exact action that was evaluated.

A proposed implementation may compute a cryptographic digest over a canonical representation of the action:

$$h_A = H(Canonicalize(A))$$

The authorization record stores the digest together with:

- Action identifier.
- Authority decisions.
- Policy version.
- Relevant context version.
- Authorization timestamp.
- Approval identity.
- Expiration or validity conditions.

Before execution, the gateway verifies that the action digest still matches.

If the action changes materially after authorization, the previous authorization must not automatically remain valid.

6.4 Time-of-Check to Time-of-Use Protection

A potential weakness occurs when the environment changes between authorization and execution.

For example, a resource may have a different owner or state when execution begins than when authorization was evaluated.

OSCA therefore proposes:

- Binding authorization to the evaluated action.
- Recording relevant environmental state.
- Using short-lived authorization decisions where appropriate.
- Revalidating critical environmental attributes immediately before execution.
- Requiring reauthorization when relevant conditions change.

This mechanism reduces the risk of executing an action under authorization conditions that no longer apply.

DOI: https://doi.org/10.5281/zenodo.23851837

7. Security and Threat Model

7.1 Threat Model

OSCA considers the following threat categories.

T1 — Unauthorized Request Origin

An attacker attempts to initiate an action using an unauthenticated or unauthorized source.

Relevant control(s): Origin Authority.

T2 — Delegation Abuse

An authorized agent attempts to delegate permissions beyond its granted authority.

Relevant control(s): Origin Authority and delegation integrity.

T3 — Semantic Scope Expansion

An agent transforms a legitimate request into a broader or materially different operation.

Relevant control(s): Semantic Authority.

T4 — Indirect Prompt Injection

An agent receives malicious instructions through untrusted external data and attempts to execute actions based on those instructions.

Relevant control(s): Origin provenance, semantic scope evaluation, and consequence authorization.

T5 — Consequence Escalation

An agent proposes an action whose effects exceed the permitted operational boundary.

Relevant control(s): Consequence Authority.

T6 — Authorization Replay

A previously authorized action is resubmitted or reused under different circumstances.

Relevant control(s): Action identifiers, expiration, context binding, and execution validation.

T7 — Post-Authorization Modification

An action is modified after authorization but before execution.

Relevant control(s): Execution binding and action digest verification.

T8 — Audit Manipulation

An attacker attempts to modify or suppress authorization records.

Relevant control(s): Protected audit storage, integrity verification, and restricted audit permissions.

7.2 Security Assumptions

The proposed architecture assumes:

1. Identity credentials and cryptographic verification mechanisms are appropriately managed.
2. Authority policies are maintained by trusted administrative processes.
3. The execution gateway cannot be bypassed by the agent.
4. Authority evaluation components have sufficiently isolated responsibilities.

5 Audit records are protected against unauthorized modification.

6 Human escalation channels are governed by explicit approval procedures.

These assumptions are important because OSCA does not independently solve every infrastructure or model-security problem.

7.3 Out-of-Scope Threats

The initial architecture does not claim to eliminate:

- All model hallucinations.
- All prompt-injection attacks.
- Compromise of the operating system or execution gateway.
- Malicious behavior by fully compromised trusted authorities.
- Incorrect organizational policies.
- Undetectable or unforeseeable consequences.

OSCA is an authorization architecture, not a complete security solution.

DOI: https://doi.org/10.5281/zenodo.23851837

8. Implementation Architecture

8.1 Design Principles

An implementation should preserve the conceptual separation of the three authority layers.

The proposed design principles are:

1. Independent authority interfaces.
2. Explicit input and output schemas.
3. Deterministic policy evaluation wherever possible.
4. Fail-closed handling of uncertainty.
5. Immutable authorization records.
6. No implicit permission inheritance.
7. Execution through a controlled gateway.

DOI: https://doi.org/10.5281/zenodo.23851837

8.2 Proposed Module Structure

```
OSCA/
■
■■■ core/
■   ■■■ action.py
■   ■■■ authorization.py
■   ■■■ decision_engine.py
■   ■■■ execution_gateway.py
■
■■■ authority/
■   ■■■ origin.py
■   ■■■ semantic.py
■   ■■■ consequence.py
■
■■■ governance/
■   ■■■ policy.py
■   ■■■ delegation.py
■   ■■■ escalation.py
■   ■■■ audit.py
■
■■■ security/
■   ■■■ action_binding.py
■   ■■■ provenance.py
■   ■■■ replay_protection.py
■
■■■ simulation/
■   ■■■ scenarios/
■   ■■■ agents/
■   ■■■ attacks/
■   ■■■ results/
■
■■■ tests/
■   ■■■ test_origin.py
■   ■■■ test_semantic.py
■   ■■■ test_consequence.py
■   ■■■ test_authorization.py
■   ■■■ test_escalation.py
■   ■■■ test_execution_binding.py
■   ■■■ test_adversarial.py
■
■■■ DOCS/
■■■ RESEARCH_PAPER/
■■■ CHANGELOG.md
■■■ README.md
■■■ VERSION
```

8.3 Authority Interface

Each authority layer should expose an independent evaluation interface.

Conceptually:

```
class AuthorityDecision:
    status: str
    reason: str
    evidence: dict

class OriginAuthority:
    def evaluate(self, action):
        ...
```

```
class SemanticAuthority:
    def evaluate(self, action):
        ...

class ConsequenceAuthority:
    def evaluate(self, action):
        ...
```

The decision engine must not replace the internal evaluation logic of any authority layer.

Its responsibility is to combine the resulting decisions according to the defined authorization policy.

8.4 Authorization Decision Algorithm

The following pseudocode represents the default strict policy:

```
def authorize(action):

    origin = origin_authority.evaluate(action)

    if origin.status == "DENY":
        return deny(action, origin)

    if origin.status != "PASS":
        return suspend_or_escalate(action, origin)

    semantic = semantic_authority.evaluate(action)

    if semantic.status == "DENY":
        return deny(action, semantic)

    if semantic.status != "PASS":
        return suspend_or_escalate(action, semantic)

    consequence = consequence_authority.evaluate(action)

    if consequence.status == "DENY":
        return deny(action, consequence)

    if consequence.status != "PASS":
        return suspend_or_escalate(action, consequence)

    return authorize_execution(action)
```

The execution gateway must independently verify the authorization record and action binding before executing the operation.

9. Experimental Methodology

9.1 Evaluation Objective

The purpose of the proposed evaluation is to determine whether OSCA's separation of authority responsibilities improves authorization accuracy, traceability, and resistance to unauthorized actions without introducing unacceptable operational overhead.

The experiment must distinguish between architectural correctness and empirical security performance.

A successful software test does not independently establish resistance to real-world adversarial attacks.

9.2 Experimental Conditions

Three principal configurations are proposed.

Configuration A — Conventional Authorization

A baseline authorization model using a unified permission evaluation mechanism.

Configuration B — Two-Dimensional Authorization

A model separating source verification from action-scope verification.

Configuration C — OSCA

The complete three-tier architecture incorporating Origin, Semantic, and Consequence Authority.

These configurations provide a comparative basis for evaluating the additional contribution of the third authority dimension.

9.3 Experimental Scenarios

The initial evaluation should include the following scenario categories.

Scenario	Description	Expected OSCA Behavior
E1	Legitimate origin and action	Authorized
E2	Unauthorized source	Origin denial
E3	Valid source, invalid action scope	Semantic denial
E4	Valid source and scope, excessive impact	Consequence denial
E5	Valid request requiring human approval	Escalation
E6	Indirect prompt injection	Unauthorized action blocked
E7	Modified action after approval	Reauthorization required
E8	Replayed authorization request	Replay rejected
E9	Conflicting authority decisions	No unauthorized execution
E10	Missing authorization evidence	Fail-closed or escalation

9.4 Adversarial Evaluation

Adversarial scenarios should include:

- Identity impersonation.
- Delegation-chain manipulation.

- Semantic scope expansion.
- Hidden instructions in retrieved content.
- Parameter substitution.
- Target substitution.
- Consequence underestimation.
- Authorization replay.
- Post-authorization action modification.
- Attempts to bypass the execution gateway.

Where possible, the evaluation should use established agent-security environments such as AgentDojo to provide realistic tool-use and indirect prompt-injection scenarios [5].

9.5 Evaluation Metrics

The following metrics are proposed.

Unauthorized Action Prevention Rate

Measures the proportion of unauthorized actions correctly prevented.

$$UAPR = (BlockedUnauthorizedActions / TotalUnauthorizedActions) \times 100$$

Legitimate Action Acceptance Rate

Measures the proportion of legitimate actions successfully authorized.

$$LAAR = (AuthorizedLegitimateActions / TotalLegitimateActions) \times 100$$

False Rejection Rate

Measures legitimate actions incorrectly denied.

$$FRR = (LegitimateActionsDenied / TotalLegitimateActions) \times 100$$

Escalation Rate

Measures the proportion of requests requiring additional approval.

$$ER = (EscalatedRequests / TotalRequests) \times 100$$

Authorization Latency

Measures the time required to complete the authorization pipeline.

$$L_A = t_{decision} - t_{request}$$

Decision Traceability Rate

Measures the proportion of authorization decisions with complete, verifiable records.

$$DTR = (CompleteDecisionRecords / TotalAuthorizationDecisions) \times 100$$

Execution-Binding Integrity

Measures the proportion of executed actions that match the exact authorized action representation.

$$EBI = (ExecutionsMatchingAuthorizedActions / TotalExecutions) \times 100$$

DOI: https://doi.org/10.5281/zenodo.23851837

9.6 Experimental Hypotheses

The following hypotheses are proposed for empirical testing.

H1: Separating Origin Authority from Semantic Authority improves the identification of actions originating from legitimate sources but exceeding their authorized scope.

H2: Adding Consequence Authority improves the detection of actions whose expected effects exceed the permitted impact boundary.

H3: Independent authority decisions improve the traceability of authorization outcomes.

H4: OSCA introduces additional authorization latency compared with a unified authorization baseline.

H5: OSCA can reduce unauthorized execution while maintaining an acceptable legitimate-action acceptance rate under defined experimental conditions.

No hypothesis is considered confirmed until supported by experimental evidence.

9.7 Experimental Controls

To improve reproducibility:

1. Use identical agent configurations across compared authorization models.
2. Use the same action corpus and scenario definitions.
3. Keep policy differences explicitly documented.
4. Separate benign and adversarial test cases.
5. Record every authorization decision.
6. Repeat stochastic agent trials.
7. Report both security and legitimate-task utility.
8. Preserve failed and incomplete executions in the dataset.
9. Report the operational cost of each authority layer independently.

DOI: https://doi.org/10.5281/zenodo.23851837

10. Expected Contributions

OSCA proposes the following contributions.

10.1 Architectural Contribution

A three-tier authorization architecture that explicitly separates source legitimacy, semantic authorization, and consequence approval.

10.2 Formal Contribution

A structured authorization model with distinct decision domains, authorization conditions, escalation rules, and architectural invariants.

10.3 Engineering Contribution

A modular implementation design that allows the three authority layers to operate independently while contributing to a unified authorization decision.

10.4 Governance Contribution

A traceable mechanism for distinguishing between the authority to initiate an action, the authority to define its permitted meaning, and the authority to approve its consequences.

10.5 Evaluation Contribution

A comparative experimental protocol for evaluating authorization accuracy, adversarial resistance, decision traceability, legitimate-action acceptance, and operational latency.

11. Discussion

11.1 Why Three Authority Layers?

The three-layer architecture is based on the observation that authorization involves different questions that should not be treated as interchangeable.

Origin Authority addresses the legitimacy of the source.

Semantic Authority addresses the legitimacy of the proposed action.

Consequence Authority addresses the legitimacy of the resulting effects.

These responsibilities are related, but they are not identical.

A legitimate source can submit an unauthorized action.

A semantically legitimate action can produce consequences requiring additional approval.

An action that is technically permitted may still exceed the authority granted for its execution.

OSCA makes these distinctions explicit within the system architecture.

11.2 Separation Without Duplication

A central design challenge is ensuring that the three authority layers remain independent without introducing unnecessary duplication.

Origin Authority should not independently redefine the semantic scope of every action.

DOI: https://doi.org/10.5281/zenodo.23851837

Semantic Authority should not assume responsibility for authenticating the original requester.

Consequence Authority should not silently replace the policy decisions of the other layers.

Instead, each layer contributes a distinct authorization decision.

The final decision engine combines these decisions without erasing their individual meanings.

11.3 Human and Institutional Authority

OSCA does not assume that every authorization decision should be delegated to an AI system.

Certain actions may require human or institutional authorization because of their impact, reversibility, legal significance, or organizational governance requirements.

The escalation mechanism provides a formal path for such decisions.

Human approval must itself be authenticated, scoped, recorded, and bound to the action under consideration.

11.4 Operational Trade-Offs

A multi-tier authorization architecture introduces additional processing steps.

These may increase latency and implementation complexity.

It may also introduce false rejections or unnecessary escalations if policies are poorly calibrated.

Consequently, OSCA should be evaluated not only through its ability to prevent unauthorized actions but also through its effect on legitimate task completion.

The security–utility trade-off must remain an explicit part of the evaluation.

11.5 Relationship to Existing Authorization Mechanisms

OSCA is intended to complement existing identity, access-control, and authorization technologies.

For example, authentication mechanisms can support Origin Authority, ABAC policies can support semantic and consequence evaluations, and fine-grained authorization protocols can represent action-specific permissions.

The proposed contribution is the architectural separation of these responsibilities within a single action authorization process.

DOI: https://doi.org/10.5281/zenodo.23851837

12. Limitations

The current work has several limitations.

First, OSCA is an architectural proposal and has not yet been empirically validated through a complete implementation and adversarial evaluation.

Second, semantic interpretation may involve uncertainty, especially when actions are generated from natural-language instructions. The architecture therefore requires structured action representations and policy-grounded evaluation rather than relying exclusively on model-generated explanations.

Third, consequence evaluation cannot guarantee the identification of every possible downstream effect. Complex systems may produce indirect or emergent consequences that cannot be fully predicted.

Fourth, the effectiveness of the architecture depends on the integrity of its identity, policy, execution, and audit infrastructure.

Fifth, the three authority layers may require domain-specific definitions. A financial transaction, a database modification, and a physical robotic action do not necessarily share identical consequence boundaries.

Finally, the architecture introduces additional operational overhead that must be measured under realistic conditions.

13. Future Research

Future work will focus on the following directions.

13.1 Formal Verification

Develop formal properties and model-checking specifications for authority separation, decision precedence, delegation integrity, and execution binding.

13.2 Executable Prototype

Implement the three authority engines, authorization decision engine, escalation manager, and execution gateway.

13.3 Adversarial Simulation

Evaluate OSCA against prompt injection, unauthorized delegation, semantic scope expansion, replay attacks, and post-authorization modification.

13.4 Domain-Specific Authorization

Evaluate the architecture in financial operations, enterprise software, autonomous research agents, and safety-critical systems.

13.5 Dynamic Consequence Evaluation

Investigate mechanisms for updating consequence assessments when environmental conditions or action effects change during execution.

13.6 Distributed Authority

Study how OSCA can operate across multiple agents, institutions, authorization services, and independently governed execution environments.

DOI: https://doi.org/10.5281/zenodo.23851837

13.7 Relationship Between Authority and Execution

Investigate whether authorization should be treated as a one-time decision or as a continuously maintained condition throughout an action's execution lifecycle.

14. Conclusion

This paper introduced OSCA, a proposed Multi-Tier Authorization Architecture for Autonomous AI Agents.

The architecture separates authorization into three independent responsibilities:

- **Origin Authority:** determining whether the source possesses legitimate authority to initiate an action.
- **Semantic Authority:** determining whether the proposed action is consistent with its authorized meaning and scope.
- **Consequence Authority:** determining whether the action's potential effects fall within an acceptable authorization boundary.

The central contribution is the explicit separation of these authority domains.

OSCA establishes that the legitimacy of an action's origin does not automatically establish the legitimacy of its meaning, and that semantic authorization does not automatically establish the authority to approve its consequences.

Through structured action representation, independent authority evaluation, explicit decision states, escalation mechanisms, and execution binding, the proposed architecture provides a foundation for implementing more granular and traceable authorization systems for autonomous AI agents.

The next stage of this research is the implementation and empirical evaluation of OSCA under controlled and adversarial conditions.

The architecture's effectiveness, operational costs, and security properties remain subjects for experimental investigation.

Design Note. *One point considered essential prior to finalizing this manuscript: Consequence Authority must be implemented as a genuinely independent authority in both design and execution — not merely as an additional risk assessment embedded within Semantic Authority. This is the most important architectural distinction that both the paper and its implementation must preserve.*

DOI: https://doi.org/10.5281/zenodo.23851837

References

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," *International Conference on Learning Representations (ICLR)*, 2023.
https://arxiv.org/abs/2210.03629

[2] V. C. Hu, D. Ferraiolo, D. R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*, NIST Special Publication 800-162, National Institute of Standards and Technology, 2019.
https://doi.org/10.6028/NIST.SP.800-162

[3] T. Lodderstedt, J. Richer, and B. Campbell, "OAuth 2.0 Rich Authorization Requests," RFC 9396, Internet Engineering Task Force, May 2023.
https://doi.org/10.17487/RFC9396

[4] J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.
https://doi.org/10.1109/PROC.1975.9939

[5] E. Debenedetti, J. Zhang, M. Balunovi■, 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 (NeurIPS)*, vol. 37, 2024.
https://doi.org/10.52202/079017-2636

[6] OWASP Foundation, *OWASP Top 10 for Large Language Model Applications 2025*, 2025.
https://genai.owasp.org/

[7] E. Tabassi, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, National Institute of Standards and Technology, 2023.
https://doi.org/10.6028/NIST.AI.100-1

Appendix A — Core Authorization Decision Table

Origin	Semantic	Consequence	Final Decision
PASS	PASS	PASS	AUTHORIZE
DENY	Any	Any	DENY
PASS	DENY	Any	DENY
PASS	PASS	DENY	DENY
ESCALATE	Any	Any	ESCALATE
PASS	ESCALATE	Any	ESCALATE
PASS	PASS	ESCALATE	ESCALATE
INDETERMINATE	Any	Any	SUSPEND
PASS	INDETERMINATE	Any	SUSPEND
PASS	PASS	INDETERMINATE	SUSPEND

Note: A DENY decision takes precedence over escalation or indeterminate states. No action is executed unless all mandatory authority layers return PASS.

Appendix B — Core OSCA Authorization Principle

$$\textit{Execution}(A) \Leftrightarrow \textit{Origin}(A) \wedge \textit{Semantic}(A) \wedge \textit{Consequence}(A)$$

Subject to:

$$\textit{ActionExecuted} = \textit{ActionAuthorized}$$

The first expression establishes the authorization condition. The second establishes the execution-binding requirement.

OSCA's foundational principle:

The authority to initiate an action, the authority to define its permissible meaning, and the authority to approve its consequences are distinct responsibilities and must remain independently verifiable.

Documentary Notes

The paper's primary references draw on published works and recognized standards: NIST for risk management and attribute-based access control, IETF for fine-grained authorization, and NeurIPS for agent-security evaluation.

Proposed Manuscript Status

Dimension	Status
Manuscript type	Architectural Research
Version	0.1.0-research-draft
Experimental status	Not yet evaluated
Figure 1	OSCA: Three-Tier Authority Separation and Action Authorization Architecture

Record ID: OSCA-ARCH-01 · Version: 0.1.0-research-draft · DOI: https://doi.org/10.5281/zenodo.23851837
Scope: Architectural proposal and formal specification · Implementation baseline: Not yet implemented.
Next Research Stage: Executable prototype development and adversarial empirical evaluation.

Samir Baladi · Independent Interdisciplinary Researcher — Ronin Institute / Rite of Renaissance · ORCID: 0009-0003-8903-0029 · gitdeeper11@gmail.com