Deontic Policies for Runtime Governance of Agentic AI Systems

Anupam Joshi
CSEE Department
UMBC
Baltimore, MD, USA
joshi@umbc.edu

Tim Finin
Center for AI
UMBC
Baltimore, MD, USA
finin@umbc.edu

Karuna Joshi
Information Systems Department
UMBC
Baltimore, MD, USA
kjoshi1@umbc.edu

Lalana Kagal
CSAIL
MIT
Cambridge, MA, USA
lkagal@csail.mit.edu

***Abstract*—Autonomous agentic AI systems driven by Large Language Models (LLMs) introduce a new class of security, privacy, and compliance challenges: an agent that can invoke tools, manipulate data, install software, and coordinate with peer agents across organizational boundaries must be constrained not just by authentication and access control, but by the full structure of enterprise governance. This includes specifying what agents are *permitted* and *prohibited* from doing, what they are *obliged* to do after certain actions (e.g., notify the CISO), under what conditions a standing obligation may be *waived*, and which rules take precedence when policies conflict. This governance problem exceeds what current policy engines provide. Systems such as XACML, Rego, and Cedar address only the permit/prohibit subset of this governance structure. They do not provide obligation lifecycle management, meta-policy conflict resolution, dispensations that waive obligations in specific circumstances, and ontological reasoning over domain class hierarchies commonly found in applications such as healthcare, cybersecurity, or data privacy. We propose AgenticRei, which realizes key governance requirements such as obligations, dispensations, policy conflict resolutions, and reasoning over policies, as well as the basic permit/prohibit constraints. We use a deontic policy language built on the Rei framework, expressed as OWL (Web Ontology Language) and evaluated at runtime by a high-performance logic engine entirely outside the LLM. The same pipeline governs both tool invocations by the agent and agent-to-agent messages. We show through examples that deontic policies capture governance constraints around security and privacy that mostly cannot be expressed in current production engines. Our approach composes naturally with industry-standard frameworks like A2AS.**

***Index Terms*—Agentic AI, policy-based governance, deontic logic, semantic web, A2AS, runtime security, multi-agent systems**

I. Introduction

The emergence of Agentic AI systems in the form of LLM-driven planners that invoke external tools, coordinate with peer agents, and execute multi-step workflows has introduced governance challenges that static, prompt-level guardrails cannot meet. A recent example using Agentic AI in a Security Operations Center identifies some of these challenges [1]. Such systems are being deployed in areas as diverse as healthcare [2] and national security [3]. The industry response has converged on a shared premise. Since LLM reasoning is non-deterministic, security must be enforced at the *point of action execution* through a deterministic policy layer independent of the LLM. Frameworks such as Wallarm's A2AS [4], Microsoft's Agent Governance Toolkit [5], and Cisco's MCP policy-enforcement gateway in Secure Access [6] all embody this premise.

This convergence validates the *where* aspect of governance: the right place to enforce policy is at the action boundary, not in the context window of the LLM. A recent joint advisory from CISA, NSA, and allied national cybersecurity agencies independently reaches the same conclusion, explicitly recommending per-invocation rather than startup-time policy evaluation [7]. But *where* to enforce is only half the problem. As we argue in Section II, the governance requirements that regulators, standards bodies, and early production deployments need go beyond what the policy languages of these frameworks can express. Our contribution addresses *the what and the how*: the expressive power of the policy language integrated with the Semantic Web, the machine-understandable layer of the Web, and fast reasoners. We discuss how this model can be integrated with industry standards like A2AS.

A2AS implements its Codified Policies (C) pillar primarily through context augmentation and the model's own reasoning, rather than through an external policy engine, consistent with its design goal of avoiding external dependencies. The A2AS authors acknowledge a resulting limitation they term *security reasoning drift*: because in-context defenses and codified policies are interpreted by the model, variations in model reasoning can lead to misinterpretation or partial compliance [4]. Providing the LLM with information about security guardrails does not guarantee compliance. Externalized engines used in industry (such as OPA [8] and Cedar [9]) avoid the LLM-as-enforcer problem, but expose a second limitation. Their policy languages are flat allow/deny or Access Based Access Control (ABAC) rules, which can only decide whether an action is permitted or denied.

We argue that open, multi-agent systems need four properties that flat-list engines cannot provide, but are captured naturally in a policy language like Rei [10] including its application to domains such as pervasive context-aware privacy [11], [12], that is based on Deontic Logic and grounded in OWL/RDF semantics (Web Ontology Language [13]). It captures the deontic notions of permissions, prohibitions, obligations, and dispensations as first-class semantic objects in ways that can then be reasoned over to make governance decisions.

© 2026, Anupam Joshi. Under license to IEEE.
DOI 10.1109/ICWS72778.2026.00159

II. Governance Gap and Case for Deontic Policies

A. The Governance Gap

Regulators, standards bodies, and early production deployments have begun to articulate what governing an autonomous agent actually requires, and the emerging consensus points well beyond access control. Three themes recur.

Oversight must scale with autonomy, and regulators are leaving the mechanism to institutions. The World Economic Forum's progressive governance model holds that the degree of oversight applied to an agent should be proportional to the autonomy it is granted [14], a position echoed by enterprise adoption surveys reporting that a large majority of executives plan agent deployment within three years. Yet the regulatory instruments that would normally supply this oversight explicitly decline to. The April 2026 interagency model-risk guidance (SR 26-2), jointly issued by the Federal Reserve, FDIC, and OCC, excludes generative and agentic AI from its scope in footnote 3, while affirming that each institution remains responsible for governing systems the guidance does not cover [15]. NIST's Center for AI Standards and Innovation has launched a dedicated AI Agent Standards Initiative to address this gap [16], signaling that agent-specific standards are still being written.

Authority creep and diffuse accountability are the dominant failure modes in practice. In production financial services, agentic systems deployed as decision-support have their confirmation requirements relaxed and autonomy thresholds raised incrementally, until a system with the action authority of a high-risk deployment is still governed as the cautious pilot it once was —"no one made a bad decision; no one made the decision at all" [17]. When such a decision is later challenged, accountability is diffuse: product owns the model, engineering owns the infrastructure, and compliance owns the policy, so no team can reconstruct who authorized the action [17]. The Five Eyes advisory identifies the technical root of this problem as non-reproducible agent decision chains, and calls for a durable record of what an agent did and under whose authority [7].

Standards say what to control, not how to enforce it. Agent certification standards such as AIUC-1 [18] define auditable control objectives—for example, control B006, "prevent unauthorized AI agent actions"—and the NIST AI Risk Management Framework [19] organizes risk management into govern, map, measure, and manage functions. Both specify *what* an enforcement layer must achieve, but are deliberately silent on the runtime mechanism. The service-infrastructure layer is left with a substantial implementation gap: how to translate a control objective written in a PDF into a decision made deterministically at every action boundary. Closing that gap is the problem addressed in AgenticRei, and we return to the concrete mapping of standards-to-runtime in Section VII.

B. Structural Requirements

A basic permission or prohibition in Rei is no more complex than its Rego or Cedar equivalent and is enforced with the same default-deny guarantee. Beyond this baseline, governance requires answering a harder question: not just whether an action is permitted, but what follows from permitting it. We identify four requirements arising from this second question in enterprise agentic deployments that are structurally inexpressible in allow/deny or ABAC rule engines.

Obligations. Real-world governance attaches not just permissions, but *duties* to permitted actions. A compliance officer who reads a regulated dataset *must* log the access within 60 seconds. A system that installs software on a production host *must* notify the CISO. These are not access-control rules; they are behavioral obligations that arise *because* access was granted. Policy languages like Rego [8] and Cedar [9] do not have a native obligation construct: the policy author must create a separate rule checking whether a prior logging event occurred, and then chain it manually. This is a workaround to the actual challenge of managing obligations.

Principled conflict resolution. In multi-organizational settings, multiple authorities issue overlapping rules. An organizational policy may prohibit exporting any dataset; a project-level policy may permit members to export a specific one; a regulatory override may permit export in response to a legal subpoena. Flat rule lists resolve such conflicts by evaluation order or an ad hoc priority integer field. Neither is semantically meaningful. Deontic frameworks provide *meta-policies* which are rules about rules. That makes conflict resolution explicit, authoritative, and auditable.

Semantic grounding. ABAC rules match the syntactic attribute values. A rule that prohibits access to "Protected Health Information (PHI)" must enumerate every concrete attribute value that constitutes PHI in the system and must be updated whenever the domain model evolves. A policy engine that can reason over a domain ontology where a "pediatric oncology record" is declared to be a subclass of "health record" which is a subclass of "PHI", can express the prohibition once, at the class level, and have it apply automatically to all current and future subclasses without touching the policy source.

Dynamic, cross-authority trust. Open agent ecosystems require that trust in a credential not just depend on *who signed it*, but that this trust relationship itself is expressed as policy. An override of an export prohibition should be granted only when the agent presents a credential whose issuer is the *compliance authority named in the policy*, and not just any credential bearing the claim "is_compliance_officer = true." This cross-pillar composition, in which the policy names trusted credential issuers and the cryptographic infrastructure delivers signed artifacts, cannot be expressed in flat ABAC rules without hard-coding issuer identifiers outside the policy. South et al. [20] extend OAuth 2.0 and OpenID Connect with agent-specific delegation credentials precisely because the base protocols, while establishing who an agent is and on whose behalf it acts, do not by themselves express which authority's credential a policy treats as trusted for a given action. This is a policy-governed trust relationship that AgenticRei makes explicit.

III. Architecture

AgenticRei enforces deontic policies at the action boundary of any agent framework. The architecture, shown in Fig. 1, follows a three-step *extract–evaluate–apply* contract. We have built a prototype that demonstrates the approach end to end. It loads Rei-encoded policies and domain ontologies, intercepts an agent action and extracts it to a ⟨subject, action, resource⟩ triple, evaluates that triple against an RDFox-based policy engine, and returns a verdict with any attached obligations. The prototype implements permission, prohibition, obligation, dispensation, and meta-policy priority resolution over the Rei ontology, with subclass reasoning supplied by RDFox's OWL/RDFS entailment. Credential verification is currently simulated through trusted-issuer matching rather than cryptographic signature checking; integration with production agent runtimes (the A2A protocol and the Microsoft Agent Framework) and with cryptographic credential verification is ongoing work.

Extract. A *TripleExtractor* intercepts every outbound action, whether by a tool invocation or an agent-to-agent (A2A) message, at the framework's middleware boundary, and maps it to a concrete ⟨*subject*, *action*, *resource*⟩ triple. This extraction happens after the agent framework has already validated the call against its schema, so the extractor operates on a typed, structured invocation, not on raw LLM text. Credentials accompanying the request are extracted in the same step and checked against the issuers the policy trusts; those that fail the check are discarded before the engine sees them.

Evaluate. A *PolicyEngine* queries a triple-store loaded with Rei-encoded deontic rules and pluggable domain ontologies. The engine returns a verdict of PERMIT, PROHIBIT, or DEFAULT-DENY, together with any obligations attached to a permitted action via the Rei `deontic:provision` construct. Every internal failure, such as an exception, timeout, or missing rule, produces DEFAULT-DENY, never an exception.

Apply. The middleware either allows execution to proceed (on PERMIT), appending obligation text to the action result for the agent to act on, or short-circuits the invocation and returns a structured policy-violation message (on PROHIBIT or DEFAULT-DENY). The LLM has no role in this decision.

Every decision is serialized to a structured audit record capturing the matched rule, round-trip latency, the credential issuers presented, and a hash of the loaded policy Knowledge Base (KB) computed from its TTL sources. The KB hash identifies which policy version was in effect for a decision, supporting forensic queries of the form "what rules were in effect when this decision was made." The record logs attribute names rather than their values, and is an append-only log rather than a tamper-evident ledger; full decision reproducibility would additionally require logging the staged request graph. Since the policy KB contains stable governance rules (not per-agent session state), the description-logic engine can materialize conclusions at load time and reduce each per-action query to a triple-pattern lookup, yielding sub-millisecond decision latency. In preliminary measurements on a single host (RHEL 9, RDFox 7.5, the policy engine reached over localhost HTTP), each prohibition/permission query pair plus any obligation lookup completes end to end in under 10 ms, with the RDFox query execution itself under a millisecond and the remainder being HTTP round-trip overhead. This is well within the bounds required for the interception of synchronous actions in production environments.

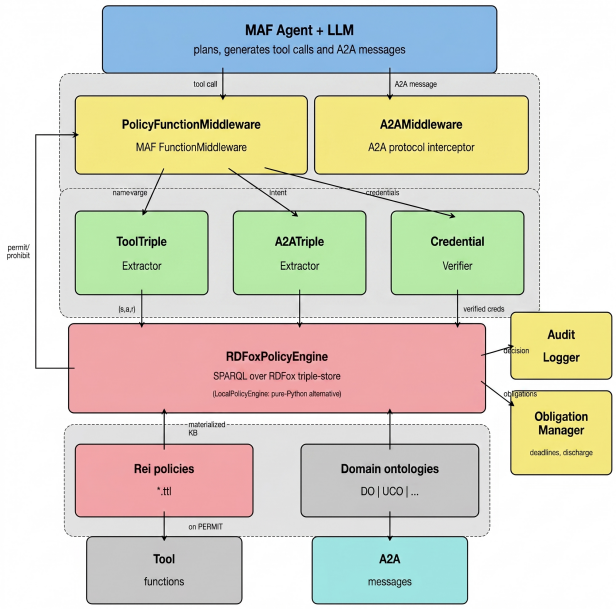

Fig. 1. AgenticRei runtime architecture. Both tool-call interception and A2A message interception share the same `PolicyEngine`, `AuditLogger`, and `ObligationManager`.

IV. Deontic Policies in Practice

The policies in AgenticRei are expressed in the Rei deontic framework [10], [11]. Rei provides four modalities directly relevant to enterprise agentic governance: *permissions*, *prohibitions*, *obligations* (attached to a permission via `deontic:provision`), and *dispensations* (releases an agent from an obligation). Meta-policies are first-class constructs: a `metapolicy:RulePriority` triple gives a project-level permission precedence over an organization-wide prohibition when both fire on the same request, replacing the evaluation-order heuristics of flat rule engines.

Domain semantics are supplied through pluggable ontology fragments loaded alongside the Rei rules. A prohibition expressed over a class (e.g., `phi:PHI`) fires automatically for any resource type declared as a subclass in the loaded domain-ontology fragment. The policy author does not need to enumerate subclasses. Adding a new ontology fragment extends the coverage of existing rules without touching the policy source.

We present the policy examples below to expose a governance property that is inexpressible in flat-list engines.

P1 — Basic Permission and Prohibition

The simplest case: any agent may read a public document; no agent may delete a user record. In Rei, permissions and prohibitions are specified using `deontic:Permission` and `deontic:Prohibition`. The rules below are instances of them and use concepts defined by the domain ontology. This can also be expressed in Rego or Cedar; P1 establishes the baseline. The prefix bindings shown below are used throughout the examples in this section; subsequent listings omit them for brevity.

```
# Rei ontology namespaces (used in all examples).
@prefix deontic <http://cs.umbc.edu/~lkagal1/rei/
    ontologies/ReiDeontic.owl#>.
@prefix constraint:<http://cs.umbc.edu/~lkagal1/
    rei/ontologies/ReiConstraint.owl#>.
@prefix metapolicy:<http://cs.umbc.edu/~lkagal1/
    rei/ontologies/ReiMetaPolicy.owl#>.
@prefix policy <http://cs.umbc.edu/~lkagal1/rei/
    ontologies/ReiPolicy.owl#>.
@prefix entity <http://cs.umbc.edu/~lkagal1/rei/
    ttlp/ReiEntity.owl#>.
@prefix action: <http://cs.umbc.edu/~lkagal1/rei/
    ontologies/ReiAction.owl#>.
@prefix rdf: <http://w3.org/1999/02/22-rdf-syntax-
    ns#>.
@prefix rdfs: <http://w3.org/2000/01/rdf-schema#>.
@prefix demo: <http://example.org/policy-demo#>.

# Domain actions are typed as Rei domain actions.
demo:ReadAction   a action:DomainAction .
demo:DeleteAction a action:DomainAction .

# variable
demo:AnyActor a entity:Variable .

demo:Perm_ReadPublicDoc a deontic:Permission ;
    deontic:actor      demo:AnyActor ;
    deontic:action     demo:ReadAction ;
    deontic:constraint demo:IsPublicDocument .

demo:Proh_DeleteUser a deontic:Prohibition ;
    deontic:actor      demo:AnyActor ;
    deontic:action     demo:DeleteAction ;
    deontic:constraint demo:IsUserRecord .

demo:BasicPolicy a policy:Policy ;
    policy:grants  demo:Perm_ReadPublicDoc ;
    policy:grants  demo:Proh_DeleteUser ;
    policy:defaultBehavior metapolicy:
    ExplicitPermImplicitProh .
```

The `policy:defaultBehavior metapolicy:ExplicitPermImplicitProh` implements the default-deny posture: anything not explicitly permitted is denied.

P2 — Obligation via `deontic:provision`

An agent may install software on a managed host, but doing so is permitted only if a CISO-notification obligation is accepted. The obligation is attached to the permission via `deontic:provision`; it fires *because* the action was permitted, not as a separate check. We are not aware of any production enforcement engine that provides this. Rego and Cedar have no obligation construct, and while the W3C's Open Digital Rights Language (ODRL) [21] defines a duty in RDF, it specifies no enforcement model or lifecycle management.

```
demo:NotifyAction          a action:DomainAction .
demo:InstallSoftwareAction a action:DomainAction .

demo:Ob_NotifyCISO a deontic:Obligation ;
    deontic:actor   demo:AnyActor ;
    deontic:action  demo:NotifyAction ;
    deontic:obligedTo demo:CISORole ;
    rdfs:comment "Notify CISO of software
    installation" .

demo:Perm_InstallSoftware a deontic:Permission ;
    deontic:actor    demo:AnyActor ;
    deontic:action   demo:InstallSoftwareAction ;
    deontic:constraint demo:IsManagedHost ;
    deontic:provision  demo:Ob_NotifyCISO .

demo:SoftwarePolicy a policy:Policy ;
    policy:grants          demo:
    Perm_InstallSoftware ;
    policy:defaultBehavior metapolicy:
    ExplicitPermImplicitProh .
```

The middleware intercepts the install action, evaluates the policy, permits execution, and surfaces the `Ob_NotifyCISO` obligation to the agent as part of the tool result. An `ObligationManager` registers the obligation with a deadline and fires an escalation callback if it is not discharged.

P3 — Policy conflicts, Meta-Policy, Cross-Pillar Composition

This example demonstrates the full power of the deontic framework and its composition with the A2AS B pillar. An organizational policy prohibits exporting any regulated dataset. An agent that presents a compliance-issued credential that is signed by the issuer *named in the policy itself* is permitted to export, because a `metapolicy:RulePriority` gives the credential-gated permission precedence over the organization-wide prohibition. An agent without such a credential, or with a credential from an untrusted issuer, is denied.

```
demo:ExportAction a action:DomainAction .

# Organization-wide prohibition
demo:Proh_ExportPII a deontic:Prohibition ;
    deontic:actor      demo:AnyActor ;
    deontic:action     demo:ExportAction ;
    deontic:constraint demo:IsRegulatedDataset .

# Trusted issuer named in the policy (B-pillar)
demo:TrustedComplianceCA a entity:Agent ;
    demo:issuerDID "did:web:acme-compliance.
    example.org" .

# Per-request credential fact: agent presents a
    credential from the named CA
demo:HasComplianceWaiver a constraint:
    SimpleConstraint ;
    constraint:subject   demo:AnyActor ;
    constraint:predicate demo:credentialIssuedBy ;
    constraint:object    demo:TrustedComplianceCA
    .

# Higher-priority permission: credential-bearing
    agents may export
```

```
demo:Perm_ExportWithApproval a deontic:Permission;
    deontic:actor      demo:AnyActor ;
    deontic:action     demo:ExportAction ;
    deontic:constraint demo:HasComplianceWaiver,
                       demo:IsRegulatedDataset .

# Meta-policy: the approved-export permission
    overrides the prohibition
demo:PriorityApprovalOverProh a metapolicy:
    RulePriority ;
    metapolicy:greaterPriority demo:
    Perm_ExportWithApproval ;
    metapolicy:lesserPriority demo:Proh_ExportPII.

demo:ExportPolicy a policy:Policy ;
    policy:grants          demo:
    Perm_ExportWithApproval ;
    policy:grants          demo:Proh_ExportPII ;
    policy:rulePriority    demo:
    PriorityApprovalOverProh ;
    policy:defaultBehavior metapolicy:
    ExplicitPermImplicitProh.
```

This policy is inexpressible in a flat-list engine for two reasons. First, the conflict between the prohibition and the permission is resolved by a meta-policy, not by evaluation order or an integer field: the `metapolicy:RulePriority` is a specified semantic object that can itself be governed by a higher-level authority. Second, the trusted issuer IRI is embedded in the policy itself, so the policy names whose signatures count, rather than trusting any attribute-bearing credential. This is the cross-pillar composition point: the B pillar delivers the signed credential; the C pillar decides whether that credential is from a trusted source.

P4 — Ontological Reasoning over Domain Class Hierarchies

The three preceding examples share a property with ABAC and flat-list engines: every constraint is expressed over a specific, named resource. In large enterprise environments, the set of regulated resource types evolves continuously, creating new entities like dataset categories, asset classifications, and record types. A policy author must enumerate each concrete type by hand, producing rules that cannot handle such evolution well. This example shows how Rei's use of the Semantic Web languages can eliminate this challenge by reasoning over type hierarchies given in domain ontologies.

A compliance policy prohibits any agent from reading Protected Health Information (PHI). In AgenticRei, this prohibition is expressed once, at the class level, against a healthcare-compliance domain ontology using the top-level class `phi:PHI`—an illustrative fragment standing in for a production ontology such as those built on HL7 FHIR or SNOMED CT. No enumeration of subtypes is needed in the policy.

```
# Domain ontology fragment (phi_fragment.ttl), not
     part of the policy itself.

@prefix phi: <http://example.org/health-compliance
    #>.

phi:PHI                  a rdfs:Class .
phi:ClinicalRecord       rdfs:subClassOf phi:PHI .
phi:PatientTreatmentPlan rdfs:subClassOf phi:
    ClinicalRecord .
phi:GeneticTestResult    rdfs:subClassOf phi:PHI .
phi:PsychiatricEvaluation rdfs:subClassOf phi:
    ClinicalRecord .
```

The Rei policy is expressed once, against the top-level class:

```
demo:Var1 a entity:Variable .
demo:AnyActor a entity:Variable .

# Constraint matches any resource whose RDF type
    is phi:PHI or any declared subclass of phi:PHI
    *; resolved by the reasoner.
demo:IsPHIResource a constraint:SimpleConstraint ;
    constraint:subject   demo:Var1 ;
    constraint:predicate rdf:type ;
    constraint:object    phi:PHI .

demo:Proh_ReadPHI a deontic:Prohibition ;
    deontic:actor      demo:AnyActor ;
    deontic:action     demo:ReadAction ;
    deontic:constraint demo:IsPHIResource .

demo:PHIPolicy a policy:Policy ;
    policy:grants          demo:Proh_ReadPHI ;
    policy:defaultBehavior metapolicy:
    ExplicitPermImplicitProh .
```

As the RDFox knowledge graph system [22] materializes the OWL subclass closure at load time, a request to read a resource typed `phi:PatientTreatmentPlan` matches `demo:IsPHIResource` without any modification to the policy, as the reasoner has already inferred that `phi:PatientTreatmentPlan` is a `phi:PHI`. The same prohibition fires for, say `phi:GeneticTestResult` and any future subclass added to the domain ontology without requiring policy modification.

In a Rego or Cedar rule set, the policy author must write an explicit condition for each concrete resource type, or maintain a separately managed allowlist that must be kept in sync with the domain model. Neither mechanism supports the open-world assumption: a resource type not in the list at authoring time is silently unprotected until a human notices and updates the rule. In AgenticRei, extending the domain ontology *is* the policy update.

P5 — Composing Prohibition, Dispensation, and Obligation: A Financial Services Example

The preceding examples each isolate one governance property. This example shows them composing in a single deployment drawn from production financial services, where agentic systems already execute transactions autonomously [17]. It directly targets the *authority creep* failure mode discussed

in Section II-A: AgenticRei makes the autonomy threshold a governed object itself. A payments agent is prohibited from autonomously executing a *high-value* transaction; a treasury-officer approval credential from the authority named in the policy provides a dispensation, and any permitted transaction that exceeds the regulatory reporting threshold carries a mandatory Currency Transaction Report (CTR) obligation under the Bank Secrecy Act. Raising the autonomy threshold is now a policy change, an action that is governed and subject to audit, rather than a silent configuration relaxation. The `TripleExtractor` stages the transaction type as an `rdf:type` assertion in the request graph, following the same pattern as P4.

As with the `phi:` fragment in P4, the `fin:` ontology below is constructed for illustration and kept deliberately small. Mature, production-grade ontologies already capture these concepts: the Financial Industry Business Ontology (FIBO) [23] models financial transactions, monetary amounts, and parties, and the Financial Regulation Ontology (FinRegOnt) [24] aligns FIBO with legal-knowledge models and a complete OWL representation of the XBRL regulatory-filing schema [25] to express US banking law and compliance filings. A production policy would be grounded in and aligned with these standards; we use a self-contained fragment here both for brevity and because the agent-action and autonomy-threshold concepts the policy governs are not native to them.

```
@prefix fin: <http://example.org/finance#> .

# TripleExtractor stages rdf:type before
    evaluation (same pattern as P4). fin:
    HighValueTransaction is a subclass of fin:
    Transaction in the domain ontology. Domain
    actions and agent are typed there.
fin:ExecutePaymentAction a action:DomainAction .
fin:FileReportAction     a action:DomainAction .
fin:PaymentsAgent        a entity:Agent .

fin:IsHighValue a constraint:SimpleConstraint ;
    constraint:subject   demo:Var1 ;
    constraint:predicate rdf:type ;
    constraint:object    fin:HighValueTransaction.

# Constraint: agent presents approval from named
    authority.
fin:TreasuryAuthority a entity:Agent ;
    fin:issuerDID "did:web:treasury-authority.
    example.org" .
fin:HasTreasuryApproval a constraint:
    SimpleConstraint ;
    constraint:subject   demo:AnyActor ;
    constraint:predicate demo:credentialIssuedBy ;
    constraint:object    fin:TreasuryAuthority .

# Prohibition: no autonomous high-value execution.
fin:Proh_AutoHighValue a deontic:Prohibition ;
    deontic:actor      fin:PaymentsAgent ;
    deontic:action     fin:ExecutePaymentAction ;
    deontic:constraint fin:IsHighValue .

# Permission overrides prohibition when treasury
    approval is present; CTR obligation fires
    BECAUSE the payment is permitted.
fin:Perm_ApprovedHighValue a deontic:Permission ;
    deontic:actor        fin:PaymentsAgent ;
    deontic:action       fin:ExecutePaymentAction;
    deontic:constraint   fin:HasTreasuryApproval ;
    deontic:provision    fin:Ob_FileCTR .

# Mandatory reporting obligation (Bank Secrecy Act
    CTR).
fin:Ob_FileCTR a deontic:Obligation ;
    deontic:actor     fin:PaymentsAgent ;
    deontic:action    fin:FileReportAction ;
    deontic:obligedTo fin:FinCEN ;
    rdfs:comment      "File CTR with FinCEN within
     15 days." .

# Dispensation: a BSA "exempt person" counterparty
    (31 CFR 1020.315)
# WAIVES the CTR filing obligation -- the agent is
    no longer obliged to file. This is an
    obligation waiver, not a prohibition override.
    fin:ExemptPerson is a counterparty class in
    the domain ontology.
fin:IsExemptCounterparty a constraint:
    SimpleConstraint ;
    constraint:subject   demo:Var1 ;
    constraint:predicate rdf:type ;
    constraint:object    fin:ExemptPerson .

fin:Disp_ExemptCTR a deontic:Dispensation ;
    deontic:actor      fin:PaymentsAgent ;
    deontic:action fin:FileReportAction ;
    deontic:constraint fin:IsExemptCounterparty .

# Meta-policy 1: approved-payment permission
    overrides the prohibition.
fin:Priority_ApprovalOverProh a metapolicy:
    RulePriority ;
    metapolicy:greaterPriority fin:
    Perm_ApprovedHighValue ;
    metapolicy:lesserPriority  fin:
    Proh_AutoHighValue .

# Meta-policy 2: an exempt counterparty waives the
    CTR obligation.
fin:Priority_ExemptOverCTR a metapolicy:
    RulePriority ;
    metapolicy:greaterPriority fin:Disp_ExemptCTR;
    metapolicy:lesserPriority  fin:Ob_FileCTR .

fin:PaymentsPolicy a policy:Policy ;
    policy:grants          fin:
    Perm_ApprovedHighValue ;
    policy:grants          fin:Proh_AutoHighValue;
    policy:grants          fin:Ob_FileCTR ;
    policy:grants          fin:Disp_ExemptCTR ;
    policy:rulePriority    fin:
    Priority_ApprovalOverProh ;
    policy:rulePriority    fin:
    Priority_ExemptOverCTR ;
    policy:defaultBehavior metapolicy:
    ExplicitPermImplicitProh.
```

This single policy exercises every property from Section II-B. The prohibition with its governed threshold addresses authority creep directly: raising the threshold is a policy change requiring re-authorization, not a silent configuration relaxation. The trusted issuer IRI is embedded in the policy itself (B-pillar composition), so the policy names whose approval counts. The CTR obligation fires *because* the transaction was permitted; a flat-list engine has no construct for a duty arising from a permitted action. Finally, the `metapolicy:RulePriority` is a named, auditable

relationship between the two rules—the evidentiary trail that resolves the diffuse-accountability problem of Section II-A by recording exactly which rule from which policy version authorized a challenged decision. The CTR obligation also connects to the obligation-lifecycle discussion in Section VII: a filing receipt from FinCEN, ingested as a W3C Verifiable Credential into the same triple-store as the obligation, converts a statutory deadline from a tracked promise to a machine-verifiable discharge record.

V. Threat Model and Scope

AgenticRei defends against *policy-violating* actions that an agent, whether through compromised reasoning, adversarial prompt injection, or deliberate misuse, attempts. This includes executing a tool call or sending an A2A message that violates governance rules. Recent overviews of attacks [26], [27] motivate the need for such strict boundary interception. Enforcement is deterministic and outside the agent. The system either invokes or permanently withholds the action, independent of what the LLM does. A prompt-injection attack that successfully causes the agent to *attempt* a prohibited action still produces a policy-violation response rather than execution.

We do not defend the agent's *reasoning process* from adversarial manipulation. Benchmarks such as AgentDojo [28], InjecAgent [29], and ASB [30] evaluate the resilience of LLM reasoning against prompt injection. This is a different property, at a different layer. The relationship is analogous to mandatory access control in an operating system: kernel policy does not prevent a user from being socially engineered to run a malicious program, but it constrains what that program can do once it runs. The two layers are complementary; neither subsumes the other. We also note that deception-based defenses [31] and mandatory policy enforcement are complementary. One targets malicious external actors, the other governs authorized internal agents.

Obligation *discharge*, the agent's subsequent behavioral act of fulfilling an issued obligation, is in an intermediate area. Obligation *issuance* is fully deterministic: when a Rei `deontic:provision` fires, the obligation is registered before the action result is returned, independent of the agent. Discharge is a behavioral property monitored via deadline tracking and post-facto audit, which are the standard governance mechanisms for obligations at organizational and legal scales. In ongoing work, we are extending the system so that obligation enforcement is automated to the extent possible and auditing is transparent.

VI. Related Work

The Rei framework [10], [32] pioneered OWL/RDF-based machine-interpretable policy reasoning and meta-policies for conflict resolution. Recent academic work extends this direction: SEAgent [33] applies Mandatory Access Control to agent-tool interactions via information-flow graphs; ShieldAgent [34] extracts verifiable rules from policy documents, represents them as linear temporal logic (LTL) constraints, and organizes them into action-based probabilistic rule circuits over which it runs probabilistic inference to verify an agent's action trajectory; and SAGA [35] provides user-controlled agent lifecycle management with a cryptographic mechanism for deriving access-control tokens that govern agent-to-agent interaction. Veriguard [36] pairs offline synthesis and formal verification of a behavioral policy with online runtime monitoring of each proposed agent action against that pre-verified policy.

Our work differs from SEAgent by replacing ABAC attribute matching with ontology-grounded deontic reasoning, and from trajectory-based approaches by providing per-action deterministic enforcement rather than global trajectory verification. Most directly comparable in recent literature are Progent [37], which enforces least-privilege tool-call control via a JSON DSL deterministically outside the LLM, AgentSpec [38], a lightweight trigger-predicate DSL for runtime safety enforcement across code, embodied, and autonomous-vehicle agents, and PCAS [39] which uses a Datalog based approach. They share our premise of deterministic per-action enforcement, but do not provide deontic obligations, meta-policy conflict resolution, or ontological reasoning. MI9 [40], developed in a financial-services context, is a broader infrastructure-agnostic runtime governance framework that combines agent-semantic telemetry, continuous authorization, finite-state-machine conformance checking, drift detection, and graduated containment; its Agency-Risk Index operationalizes the principle of Section II-A that oversight should scale with autonomy. Its conformance engine matches temporal event patterns rather than evaluating deontic rules, so like the trajectory-based approaches, it expresses neither obligations and dispensations nor ontology-grounded, meta-policy conflict resolution. The two are complementary, as MI9's telemetry and drift detection could supply the behavioral signals that trigger AgenticRei policy evaluation.

On the industry side, the A2AS BASIC model [4] is the most directly comparable framework. Our work is an alternative realization of its Codified Policies pillar that externalizes policy evaluation while consuming the Behavior pillar's cryptographic constructs. OPA [8] and Cedar [9] also provide externalized policy engines but lack obligations, dispensations, meta-policies, and ontological reasoning. The W3C Open Digital Rights Language [21] is structurally closer to Rei, but lacks a runtime enforcement architecture, obligation lifecycle management, and meta-policy conflict resolution. The CoSAI/OASIS secure design patterns [41] explicitly recommend OPA, Cedar, and OpenFGA as policy languages; AgenticRei is a richer, more expressive alternative in that space.

PONDER [42] established a principle we follow: policy is specified declaratively and independently of the mechanism that enforces it, so policy can change without changing the underlying implementation, with reusable role- and relationship-based structures for large organizations, although PONDER's focus is object-oriented systems. The complementary separation of a stable policy decision point from the per-request enforcement point is XACML's PEP/PDP pattern, on which AgenticRei builds [43], extended by Bertino et al. for dynamic

context and derived-data control [44], [45], but addresses three gaps. First, XACML obligations are PEP-directed annotations without lifecycle management rather than deontic duties derived from permissions; second, conflict resolution uses per-PolicySet combining algorithms rather than named, governable meta-policies; and finally, attribute matching cannot be grounded in OWL class hierarchies. OPA [8] resolves conflicts through bundle-load ordering external to the policy language; Cedar [9] through a fixed `forbid`-overrides-`permit` default; neither supports named, directed priority relationships between specific rules that are first-class policy objects governable by a higher authority, as Rei provides.

The Belnap policy logic program [46], [47] also rejects binary permit/deny and provides formal static conflict-detection analysis. The approaches are complementary: Belnap logic for pre-deployment compositional analysis, AgenticRei for runtime deontic governance. AOPL-P [48] extends an authorization-and-obligation policy language with penalties and answer-set-programming-based planning, letting agents weigh the penalty of non-compliance against goal utility and, in high-stakes situations, deliberately select a policy-violating plan.

The governance standards that motivate this work—AIUC-1 [18], the NIST AI Risk Management Framework [19], and the NIST Generative AI Profile [49]—are discussed in Section II-A; they define control objectives but are mechanism-agnostic, and AgenticRei provides a runtime realization of their action-governing subset (Section VII). Two further bodies of work are complementary rather than competing. Threat taxonomies such as MITRE ATLAS [50] and the IBM AI Risk Atlas [51] catalog adversarial techniques and risk categories for AI systems; AgenticRei constrains the impact stage of an ATLAS-style attack chain regardless of whether an upstream technique such as prompt injection succeeds, complementing the detection-oriented defenses these taxonomies motivate. Frontier model safety policies such as Anthropic's Responsible Scaling Policy [52] operate at the training-time, capability-threshold level, and address catastrophic model-level risk; they govern *which models may be deployed*, a concern orthogonal to AgenticRei's governance of *what actions a deployed agent may take*.

VII. Vision and Open Challenges

Earlier in the paper (Section II-A), we identified the governance gaps facing agentic AI systems and our current attempts to use Deontic Logic-based Security Governance frameworks in closing them. The examples in Section IV show what a deontic language can express today. We next outline what remains. Closing the standards-to-runtime gap programmatically, from a control objective written in a PDF to a decision made deterministically at every action boundary, is the central challenge AgenticRei is designed to address, and one that Koch [53] identifies as an open frontier.

Federated policy delegation. In open agent ecosystems where new agent types are deployed continuously, policy bases must evolve without requiring the full rule set to be redeployed. Rei's OWL representation makes incremental policy update tractable: adding a new ontology fragment or a new Rei rule at runtime changes only the relevant portion of the materialized KB, not the full closure. The harder open problem is a formal protocol for *delegated* policy update, in which a sub-authority can extend but not override a parent authority's rules. This is the structural fix for the *authority creep* failure mode of Section II-A. When the autonomy threshold is a governed object in the policy KB (as in P5), raising it becomes a policy update that a delegated-update protocol can require a parent authority to authorize, rather than a silent configuration change that no one is accountable for.

Realizing governance standards at runtime. Section II-A observed that agent standards specify what to control but not how to enforce it. AgenticRei provides a deterministic enforcement substrate for the subset of controls expressible as deontic constraints over agent actions. As shown in [54], [55], these can capture security constraints important in the industry, such as mandatory access control, role based access control, and attribute based access control. For instance, AIUC-1's B006 (prevent unauthorized agent actions) maps to Rei prohibitions under default-deny. Similarly, D (restrict unsafe tool calls) maps to prohibitions over tool-invocation action classes, and A (limit data access, prevent cross-tenant exposure) maps to resource constraints with ontological grounding. Another instance is C (flag high-risk actions for human review) mapping to the credential-gated override pattern as in P3 and P5, where human-issued credentials gate permission. Finally, E (log AI activity, assign accountability) maps to the structured, policy-versioned audit record. Demonstrating conformance to the action-governing subset of AIUC-1 or the NIST AI RMF MANAGE function is achieved in part by deploying the corresponding Rei policies, with the audit record serving as machine-verifiable evidence for an auditor. This operationalizes the standards-to-runtime translation that Koch [53] frames as an open challenge. We pair the enforcement layer (Rei policies) with the evidence layer (Verifiable Credential discharge records, discussed next).

Obligation lifecycle governance. The current framework issues obligations deterministically and tracks their deadlines. When the obligation is explicitly given a dispensation, that can be handled as well. However, the question of what *evidence* constitutes discharge and how that evidence is verified requires a richer model. Linking obligations to W3C Verifiable Presentations [56] closes this loop: the CISO system issues a VC attesting that the notification was received; the VC is ingested into the same triple-store as the obligation; and a single SPARQL query can then answer "show me every obligation issued this month, its deadline, and its discharge credential." This also works on agents cooperating across organizational boundaries, as long as there is a common root of cryptographic trust. This transforms obligation tracking from a best-effort audit trail into a machine-verifiable governance record, directly addressing the non-reproducible decision chains identified in Section II-A. The CTR obligation in P5 illustrates the pattern

concretely: a filing receipt from FinCEN, ingested as a VC, converts a statutory deadline into an auditable governance record. We are incorporating this into ongoing work.

Policy authoring at scale. Expressive policies are only useful if practitioners can author and maintain them without a specialist in both governance and formal logic on every team. Policy engineering is a challenge that requires experts in both governance and logic. Translating natural-language governance requirements into policies expressed in Rei, perhaps using an LLM, with formal verification of the output, would lower the barrier to adoption. Checking to make sure that the policies are good [36] and conflict free (e.g., [57]) provides the static analysis complement that Rei currently lacks. Recent work such as Policy-as-Prompt [58] demonstrates the authoring half of this pipeline, automatically extracting a structured, source-linked policy tree from unstructured design artifacts such as product and design documents; it compiles the result, however, into prompt-based LLM classifiers acting as runtime judges, reintroducing the very non-determinism that externalized engines are designed to avoid. AgenticRei can consume such an extraction front-end while preserving deterministic, LLM-independent enforcement and the obligation and meta-policy constructs that prompt classifiers cannot represent. The World Economic Forum's *agent card* concept, a structured pre-deployment documentation of an agent's authorized capabilities and governance constraints [14], is a natural upstream artifact from which Rei policy templates could be derived. Combining tools for policy authoring and quality control with Rei-based runtime governance is an important direction for our future work.

VIII. Conclusion

We have argued that the full governance requirements of open Agentic AI systems relating to security, privacy, and compliance are beyond what current policy engines can express. We identified some needed constructs and argued that a deontic policy language like Rei provides them. AgenticRei demonstrates that such policies can be evaluated at the execution time of the action, entirely outside the LLM, with low-millisecond latency. Preliminary measurements show sub 10-ms end-to-end latency per decision, dominated by the HTTP request time, well within the bounds required for synchronous action interception. AgenticRei works synergistically with the cryptographic credential infrastructure of frameworks such as A2AS. The key design insight is the separation of two concerns that flat-list engines conflate: *what* an agent is allowed to do (permission), and *what follows* from allowing it (obligation). Deontic logic makes this separation explicit. Enforcing it deterministically, at scale, in production agentic systems is the challenge we are working toward.

Acknowledgments

This work was partially funded by NSF award 2310844, IUCRC Phase II UMBC: Center for Accelerated Real-Time Analytics (CARTA). Generative AI systems were used to create the structure of the document, curate related work, check grammar, and refine the presentation. They were also used to help code the policy enforcement prototype.

References

[1] K. Banstola, F. Al Faisal, and X. Ou, "Experiences of using agentic AI to fill tooling gaps in a security operations center," in *Workshop on SOC Operations and Construction (WOSOC), co-located with NDSS 2026*, 2026. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/wosoc26-02.pdf

[2] C. H. Savage, A. Kanhere, V. Parekh, C. P. Langlotz, A. Joshi, H. Huang, and F. X. Doo, "Open-source large language models in radiology: a review and tutorial for practical research and clinical deployment," *Radiology*, vol. 314, no. 1, p. e241073, 2025.

[3] S. S. L. Chukkapalli, D. Ledbetter, A. Joshi, T. Finin, and J. Freeman, "Impostors among us: An agentic approach to identifying and resolving conflicts in collaborative network environments," in *IEEE Military Communications Conference*, 2025.

[4] E. Neelou, I. Novikov, M. Moroz, O. Narayan, T. Saade, M. Ayenson, I. Kabanov, J. Ozmen, E. Lee, V. S. Narajala, E. Guilherme Junior, K. Huang, H. Gulsin, J. Ross, M. Vyshegorodtsev, A. Travers, I. Habler, and R. Jadav, "A2AS: Agentic ai runtime security and self-defense," 2025. [Online]. Available: https://arxiv.org/abs/2510.13825

[5] Microsoft Open Source, "Introducing the agent governance toolkit: Open-source runtime security for AI agents," https://opensource.microsoft.com/blog/2026/04/02/introducing-the-agent-governance-toolkit-open-source-runtime-security-for-ai-agents/, 2026.

[6] Cisco, "Cisco reimagines security for the agentic workforce," https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2026/m03/cisco-reimagines-security-for-the-agentic-workforce.html, 2026.

[7] ASD's ACSC, CISA, NSA, Canadian Centre for Cyber Security, NCSC-NZ, and NCSC-UK, "Careful adoption of agentic AI services," United States Cybersecurity and Infrastructure Security Agency, Tech. Rep., April 2026. [Online]. Available: http://bit.ly/49JP3wm

[8] Styra, "Open policy agent (opa) – philosophy and rego language reference," 2025. [Online]. Available: https://www.openpolicyagent.org/docs/

[9] J. W. Cutler, C. Disselkoen, A. Eline, S. He, K. Headley, M. Hicks, K. Hietala, E. Ioannidis, J. Kastner, A. Mamat, D. McAdams, M. McCutchen, N. Rungta, E. Torlak, and A. M. Wells, "Cedar: A new language for expressive, fast, safe, and analyzable authorization," in *Proceedings of OOPSLA*, 2024, pp. 670–697.

[10] L. Kagal, T. Finin, and A. Joshi, "A policy language for a pervasive computing environment," in *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. IEEE, 2003, pp. 63–74.

[11] H. Chen, T. Finin, and A. Joshi, "Semantic web in a pervasive context-aware architecture," in *Workshop on Artificial Intelligence in Mobile System, 5th Annual Conf. on Ubiquitous Computing*, A. Krüger and R. Malaka, Eds. IEEE, 2003, pp. 33–40.

[12] A. Patwardhan, V. Korolev, L. Kagal, and A. Joshi, "Enforcing policies in pervasive environments," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004*, 2004, pp. 299–308.

[13] World Wide Web Consortium, "Owl - semantic web standards - w3c," https://www.w3.org/OWL/, 2026, accessed: 2026-06-12.

[14] World Economic Forum and Capgemini, "AI agents in action: Foundations for evaluation and governance," Nov. 2025. [Online]. Available: https://reports.weforum.org/docs/WEF_AI_Agents_in_Action_Foundations_for_Evaluation_and_Governance_2025.pdf

[15] Board of Governors of the Federal Reserve System, Federal Deposit Insurance Corporation, and Office of the Comptroller of the Currency, "Supervisory guidance on model risk management," Board of Governors of the Federal Reserve System, Tech. Rep. SR Letter 26-2, Attachment, Apr. 2026, footnote 3 explicitly excludes generative and agentic AI models from scope; institutions remain responsible for their governance.

[16] National Institute of Standards and Technology, "AI agent standards initiative, center for AI standards and innovation (CAISI)," 2026. [Online]. Available: https://www.nist.gov/artificial-intelligence/ai-agent-standards-initiative

[17] R. Saxena, "Agentic AI governance lessons from financial services," Responsible AI Institute; republished in the PLI Chronicle, Apr. 2026. [Online]. Available: https://www.responsible.ai/news/agentic-ai-governance-lessons-from-financial-services/

[18] AIUC-1 Consortium, "AIUC-1: A certification standard for AI agents," 2026, six control families: data/privacy, security, safety, reliability, accountability, and society; crosswalks to NIST AI RMF, MITRE ATLAS, ISO 42001, and OWASP agentic lists. [Online]. Available: https://www.aiuc-1.com/

[19] National Institute of Standards and Technology, "Artificial intelligence risk management framework (AI RMF 1.0)," National Institute of Standards and Technology, Tech. Rep. NIST AI 100-1, 2023.

[20] T. South, S. Marro, T. Hardjono, R. Mahari, C. D. Whitney, D. Greenwood, A. Chan, and A. Pentland, "Authenticated delegation and authorized AI agents," 2025. [Online]. Available: https://arxiv.org/abs/2501.09674

[21] W3C ODRL Community Group, "ODRL information model 2.2," W3C, 2018. [Online]. Available: https://w3.org/TR/odrl-model/

[22] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu, and J. Banerjee, "Rdfox: A highly-scalable rdf store," in *The Semantic Web - ISWC 2015*. Springer International Publishing, 2015.

[23] EDM Council, "Financial industry business ontology (FIBO)," https://edmcouncil.org/frameworks/industry-models/fibo/, 2024.

[24] Jayzed Data Models Inc., "Financial regulation ontology (FinRegOnt): Semantic compliance in finance," https://finregont.com/, 2024.

[25] ——, "The financial regulation XBRL ontology," https://finregont.com/xbrl-ontology/, 2024.

[26] S. Puppala, I. Hossain, M. J. Alam, Y. Lee, J. Yoo, T. Ahad, S. B. Alam, and S. Talukder, "Agent-fence: Mapping security vulnerabilities across deep research agents," arxiv:2602.07652, 2026. [Online]. Available: https://arxiv.org/abs/2602.07652

[27] Z. Deng, Y. Guo, C. Han, W. Ma, J. Xiong, S. Wen, and Y. Xiang, "AI agents under threat: A survey of key security challenges and future pathways," *ACM Computing Surveys*, vol. 57, no. 7, 2025.

[28] 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," in *Advances in Neural Information Processing Systems 37 (NeurIPS 2024) Datasets and Benchmarks Track*, 2024. [Online]. Available: https://openreview.net/forum?id=m1YYAQjO3w

[29] Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in *Findings of the Association for Computational Linguistics*. Bangkok, Thailand: ACL, 2024, pp. 10 471–10 506. [Online]. Available: https://aclanthology.org/2024.findings-acl.624

[30] H. Zhang, J. Huang, K. Mei, Y. Yao, Z. Wang, C. Zhan, H. Wang, and Y. Zhang, "Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents," in *13th Int. Conference on Learning Representations)*, 2025. [Online]. Available: https://openreview.net/forum?id=KbwJPDKimp

[31] D. Ayzenshteyn, R. Weiss, and Y. Mirsky, "Cloak, honey, trap: Proactive defenses against LLM agents," in *Proceeddings of the 34th USENIX Security Symposium*. Seattle, WA, USA: USENIX Association, 2025, pp. 8095–8114. [Online]. Available: https://usenix.org/conference/usenixsecurity25/presentation/ayzenshteyn

[32] L. Kagal, T. Finin, A. Joshi, and S. Greenspan, "Security and privacy challenges in open and dynamic environments," *IEEE Computer*, vol. 39, no. 6, pp. 89–91, Jun. 2006.

[33] Z. Ji, D. Wu, W. Jiang, P. Ma, Z. Li, Y. Gao, S. Wang, and Y. Li, "Taming various privilege escalation in llm-based agent systems: A mandatory access control framework," *arXiv preprint arXiv:2601.11893*, 2026.

[34] Z. Chen, M. Kang, and B. Li, "Shieldagent: Shielding agents via verifiable safety policy reasoning," *arXiv preprint arXiv:2503.22738*, 2025.

[35] G. Syros, A. Suri, J. Ginesin, C. Nita-Rotaru, and A. Oprea, "SAGA: A security architecture for governing AI agentic systems," in *Network and Distributed System Security (NDSS) Symposium*, 2026. [Online]. Available: https://arxiv.org/abs/2504.21034

[36] L. Miculicich, M. Parmar, H. Palangi, K. D. Dvijotham, M. Montanari, T. Pfister, and L. Le, "Veriguard: Enhancing llm agent safety via verified code generation," 2025.

[37] T. Shi, J. He, Z. Wang, L. Wu, H. Li, W. Guo, and D. Song, "Progent: Programmable privilege control for LLM agents," *arXiv preprint arXiv:2504.11703*, 2025. [Online]. Available: https://arxiv.org/abs/2504.11703

[38] H. Wang, C. M. Poskitt, and J. Sun, "AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents," in *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE 2026)*, 2026. [Online]. Available: https://arxiv.org/abs/2503.18666

[39] N. Palumbo, S. Choudhary, J. Choi, P. Chalasani, M. Christodorescu, and S. Jha, "Policy compiler for secure agentic systems," 2026.

[40] C. L. Wang, T. Singhal, A. Kelkar, and J. Tuo, "MI9 — agent intelligence protocol: Runtime governance for agentic AI systems," *arXiv preprint arXiv:2508.03858*, 2025.

[41] CoSAI / OASIS Workstream 4, "Model context protocol (MCP) security: Secure design patterns for agentic systems," https://github.com/cosai-oasis/ws4-secure-design-agentic-systems, 2025.

[42] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder policy specification language," in *Policies for Distributed Systems and Networks (POLICY 2001)*, ser. Lecture Notes in Computer Science, vol. 1995. Berlin, Heidelberg: Springer, 2001, pp. 18–38.

[43] OASIS, "eXtensible Access Control Markup Language (XACML) version 3.0," OASIS Standard, 2013. [Online]. Available: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[44] N. Ammar, Z. Malik, E. Bertino, and A. Rezgui, "XACML policy evaluation with dynamic context handling," *IEEE Transactions on Knowledge and Data Engineering*, 2015.

[45] E. Bertino, D. Verma, and S. Calo, "A policy system for control of data fusion processes and derived data," in *21st International Conference on Information Fusion*, 2018, pp. 807–813.

[46] C. Hankin, F. Nielson, and H. R. Nielson, "Advice from Belnap policies," in *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF 2009)*. IEEE, 2009, pp. 234–247.

[47] G. Bruns and M. Huth, "Access control via Belnap logic: Intuitive, expressive, and analyzable policy composition," *ACM Transactions on Information and System Security*, vol. 14, no. 1, pp. 9:1–9:27, 2011.

[48] V. Tummala and D. Inclezan, "Autonomous agents and policy compliance: A framework for reasoning about penalties," *arXiv preprint arXiv:2512.03931*, 2025, under consideration in Theory and Practice of Logic Programming (TPLP).

[49] National Institute of Standards and Technology, "Artificial intelligence risk management framework: Generative artificial intelligence profile," National Institute of Standards and Technology, Tech. Rep. NIST AI 600-1, Jul. 2024.

[50] MITRE, "ATLAS: Adversarial threat landscape for artificial-intelligence systems," https://atlas.mitre.org/, 2025.

[51] IBM, "AI risk atlas," 2025. [Online]. Available: https://www.ibm.com/docs/en/watsonx/saas?topic=ai-risk-atlas

[52] Anthropic, "Responsible scaling policy, version 3.0," 2026, effective February 24, 2026. [Online]. Available: https://www.anthropic.com/responsible-scaling-policy

[53] C. Koch, "From governance norms to enforceable controls: A layered translation method for runtime guardrails in agentic ai," arxiv:2604.05229, 2026. [Online]. Available: https://arxiv.org/abs/2604.05229

[54] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham, "R owl bac: representing role based access control in owl," in *Proceedings of the 13th ACM symposium on Access control models and technologies*. ACM, 2008, pp. 73–82.

[55] N. K. Sharma and A. Joshi, "Representing attribute based access control policies in owl," in *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*. IEEE, 2016, pp. 333–336.

[56] W3C, "Verifiable credentials data model 2.0," W3C Recommendation, 2024.

[57] E. Bertino, A. Abu Jabal, S. Calo, C. Makaya, M. Touma, D. Verma, and C. Williams, "The challenge of access control policies quality," *ACM Journal of Data and Information Quality*, vol. 10, no. 2, pp. 6:1–6:6, 2018.

[58] G. Kholkar and R. Ahuja, "Policy-as-prompt: Turning AI governance rules into guardrails for AI agents," in *Workshop on Regulatable ML, 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025. [Online]. Available: https://arxiv.org/abs/2509.23994