ArgorixLang: Compiled Governance, Sovereign Agent Metadata, and Offline-Verifiable Runtime Evidence

Gustavo Venegas*, Edison Vazquez*, Danilo Naranjo†, Benjamin Gonzalez*
*Chilean Chamber of Artificial Intelligence †Ocular
gustavo@gobiernoia.cl, edison@jhedai.com, dan@ocularsolution.com, ben@nirvana-ai.com

***Abstract*—Agent runtimes mix identity metadata, provider permissions, policy decisions, and audit evidence, but these layers are often distributed across prompts, configuration, application code, and logs. This paper presents ArgorixLang as a systems artifact for *compiled governance*: a language/runtime boundary in which agent declarations, passports, provider contracts, executable-provider permissions, runtime controls, and evidence artifacts are compiled and checked together. The paper deliberately separates declaration from verification, digest integrity from approval, local passport metadata from authenticated identity, provider contracts from executable adapters, and EvidenceBundles from external attestation. We evaluate the current artifact through a reproducible snapshot of 33 generated request directories. Twenty-seven directories contain complete bytecode, trace, security-report, and EvidenceBundle artifacts; six contain source only. The 27 complete bundles pass offline internal-consistency verification for bytecode, trace, security report, and a ledger digest recomputed from trace events. The snapshot is structurally repeated, so its 7,155 event records and 1,188 detailed policy findings are evidence of deterministic artifact production rather than workload diversity. A negative finding is central: all complete v0.1 policy reports require review and encode their detailed findings as "unknown policy rule," showing that digest success does not imply policy approval. We therefore evaluate a controlled v0.2 extension: a typed policy decision lattice with pass, deny, review, unknown-rule, and malformed-policy outcomes, plus source-digest binding in a controlled local matrix. The generated matrix covers 31 national country codes and EU as a regional residency zone, producing 57 deterministic cases for passport metadata, residency conflicts, provider boundaries, tamper detection, and policy diagnostics. These cases test local metadata handling and evidence consistency only. The current paper claims a compiled governance and offline evidence boundary; it does not claim security certification, real identity verification, live ANS/DNS, cryptographic attestation, production sandboxing, legal compliance, physical data residency, or large-scale empirical coverage.**

***Index Terms*—agent programming languages, compiled governance, sovereign identity, runtime evidence, fail-closed execution, reproducibility**

I. Introduction

Modern agent systems compose identities, tools, model providers, delegated permissions, jurisdictional claims, policy decisions, and evidence artifacts. In many deployments those layers live in different places: prompts describe intent, configuration names providers, application code enforces some checks, logs record partial outcomes, and documents describe governance after the fact. This separation is operationally convenient, but it makes a systems question hard to answer: what can be represented, restricted, and evidenced together before an agent operation reaches an executable provider?

ArgorixLang answers this question as a compiled governance runtime rather than as a certification framework. It compiles agent declarations, typed messages, local Agent Passport metadata, provider contracts, policies, runtime profiles, and evidence requirements into bytecode that is executed by a fail-closed VM [1]. The VM separates a declarative provider contract from an executable provider adapter. A contract may describe an external provider; it does not become callable unless an executable adapter is registered. In the artifact studied here, simulated is the only executable provider.

The central thesis is claim-bounded: ArgorixLang demonstrates a language/runtime boundary that can compile governance metadata, deny unsupported provider execution, and generate internally verifiable local evidence. It does not turn declarations into external truth. Throughout the paper we enforce five separations:

$$\begin{aligned} \text{declaration} &\neq \text{verification}, \\ \text{digest integrity} &\neq \text{policy approval}, \\ \text{local passport metadata} &\neq \text{authenticated identity}, \\ \text{provider contract} &\neq \text{executable adapter}, \\ \text{EvidenceBundle} &\neq \text{external attestation}. \end{aligned}$$

These separations are not caveats appended at the end; they define the method.

Related systems occupy adjacent points in the design space. MCP and A2A standardize agent interaction surfaces [2, 3]. NANDA and AgentFacts motivate open naming and discovery for agents [4, 5]. DID and VC standards define identifier and credential data models [6, 7]. OPA and Cedar demonstrate policy-as-code approaches [8, 9]; SPIFFE/SPIRE address workload identity [10, 11]; in-toto and SLSA address provenance and supply-chain evidence [12, 13]. ArgorixLang does not replace these systems. Its contribution is to compile local governance constructs and evidence boundaries in one agent runtime while marking unresolved assurance properties as unresolved.

The paper asks five explicit research questions:

RQ1. How can governance, sovereign identity, and evidence requirements be represented as compiled language constructs?

RQ2. How does the ArgorixLang pipeline constrain a chatbot request before a provider operation can be planned?
RQ3. What evidence does the runtime generate, and how can its internal consistency be checked offline?
RQ4. What do the existing runtime sessions demonstrate, and which controlled experiments remain required?
RQ5. How can Agent Passports, ATrust, DCP-AI, and future sovereign discovery fit into an open agentic-web architecture?

Our contributions are:

1) **Compiled governance constructs:** a language architecture that represents agents, passports, policies, runtime profiles, provider contracts, and evidence requirements in one validated pipeline.
2) **Fail-closed provider boundary:** a split between declarative provider contracts and executable adapters, with unsupported external-provider execution denied before provider planning.
3) **Typed policy lattice:** a v0.2 controlled-matrix policy model separating pass, deny, review, unknown-rule configuration errors, and malformed policy objects instead of collapsing unknown rules into ordinary violations.
4) **Offline evidence boundary:** an EvidenceBundle model that checks bytecode, trace, security-report, and ledger digest consistency while explicitly excluding source integrity, authorship, timestamp, and side-effect absence from the current claim.
5) **Controlled evaluation agenda:** a snapshot analysis plus generated controlled-case matrix, ablations, baselines, and claim-boundary taxonomy.

To prevent overreading, we classify every claim into one of four categories from the outset, summarized in table III and applied uniformly thereafter. *Implemented* mechanisms include the compiler, the fail-closed VM path, the simulated-only provider boundary, traces, the SecurityReport, the EvidenceBundle, and semantic digest verification. *Declarative* metadata includes passports, ans_name, ATrust maps, MCP and A2A bridge declarations, and external-provider contracts. *Proposed* architecture includes Sovereign DNS/ANS resolution, a DID/VC resolver, signed bundles, source-digest binding, and live federation. We explicitly do *not* claim real identity authentication, credential verification, cryptographic handshakes, blockchain immutability, post-quantum security, legal-compliance certification, universal prompt-injection prevention, or production isolation.

Figure 1 summarizes the stack. The empirical result is deliberately narrower than an assurance claim: we observe deterministic structure, bounded execution, and verifiable artifact relationships in one repeated snapshot. We neither conduct a controlled attack experiment nor infer production security from successful digest verification.

Where a figure and table address the same topic, they are deliberately paired rather than duplicative: the figure communicates topology, flow, or boundary placement, while the table provides exact categorical or quantitative comparison. Captions and prose identify the distinct evidentiary role of each.

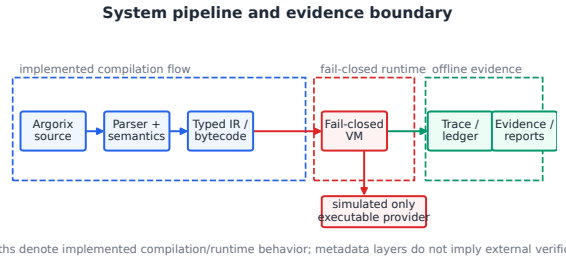

Fig. 1. ArgorixLang compilation and evidence pipeline. Solid paths denote implemented compilation and runtime behavior. Metadata layers do not imply external verification, live federation, legal compliance, or real-world identity proof.

II. Background and Related Work

A. Agent naming and discovery

Project NANDA's current official title is *Architecting the Internet of AI Agents* [4]. Its research anchor, "Beyond DNS: Unlocking the Internet of AI Agents via the NANDA Index and Verified AgentFacts," proposes indexing and verifiable facts for networked agents [5]. DNS itself separates hierarchical naming from application semantics [14, 15]. ArgorixLang does not implement NANDA, AgentFacts publication, ANS federation, or remote DNS resolution. The comparison is architectural: locally compiled sovereign metadata could become an input to a future discovery service.

B. Authenticated delegation and agent authorization

OAuth/OIDC-based agent delegation drafts, FIDO's Agentic Authentication Working Group, Google AP2, and Mastercard Verifiable Intent all address how agents should request, delegate, authenticate, or evidence user intent in real ecosystems [16–19]. ArgorixLang does not implement these flows. It can declare authorization and credential requirements locally, but it does not authenticate a human, bind a payment mandate, or verify a delegation token.

C. Non-human and workload identity

DID Core and the Verifiable Credentials Data Model define identifier and credential representations [6, 7]. SPIFFE/SPIRE define workload identity mechanisms for software systems [10, 11], and OWASP's Non-Human Identities Top 10 catalogs operational risks for service accounts, keys, and machine principals [20]. ArgorixLang passports are local metadata: declaring an identifier or country field is not equivalent to resolving, issuing, or verifying identity.

D. Policy-as-code and enforcement

Open Policy Agent separates a general-purpose policy decision engine from application enforcement and expresses decisions in Rego [8]. Cedar offers a policy language oriented around authorization decisions [9]. ArgorixLang instead compiles policy together with agent, passport, provider, and evidence declarations and emits VM-local decision events. It is not an OPA/Rego or Cedar evaluator.

TABLE I
SCOPE OF RELATED-WORK COMPARISONS. EACH ROW IS RELATIONAL, NOT AN EQUIVALENCE OR CONFORMANCE CLAIM.

System	Comparison dimension	Bibliographic scope
Project NANDA / AgentFacts / ANS	Agent naming and discovery	Project-level comparison only
ATrust	Agent trust relationships	Project-level comparison only
DCP-AI	Discovery and control plane	Project-level comparison only
OAuth/OIDC, FIDO, AP2, Verifiable Intent	Delegation and agent authorization	Conceptual comparison only
SPIFFE/SPIRE and OWASP NHI	Non-human/workload identity	Scope comparison only
Open Policy Agent	External policy-as-code engine	Direct architectural comparison
Cedar	Authorization policy language	Conceptual comparison only
Jason / AgentSpeak	BDI agent language and interpreter	Direct language/runtime comparison
in-toto / SLSA / VET Your Agent	Evidence, provenance, and external evaluation	Scope comparison only
WASI	Capability-based host isolation	Direct isolation-boundary comparison

E. Evidence, provenance, and attestation

in-toto records and verifies software supply-chain steps [12]; SLSA defines supply-chain assurance levels and provenance expectations [13]; VET Your Agent studies agent security evaluation as an external assessment problem [21]. ArgorixLang EvidenceBundles concern one local runtime result, use unsigned semantic digests, and currently exclude source. They do not provide signer authentication, trusted time, TEE attestation, zkTLS/web-proof evidence, or replacement-resistant transparency.

F. Agent protocols and capability-based isolation

MCP defines a protocol for context and tool interactions; A2A defines inter-agent communication [2, 3]. WASI specifies standards-track host APIs and a capability-based sandbox model [22]. ArgorixLang models bridge constraints and records negative execution controls, but it does not claim live MCP/A2A conformance or WASI containment in the current artifact.

Table I pairs each system with one comparison dimension. Its rows are side-by-side scope comparisons; neither the table nor the surrounding text claims implementation equivalence, protocol conformance, or shared assurance.

The language is implemented in Rust [23]. The evolution in the repository reflects a sequence from agent execution toward typed messages, provider contracts, passports, ATrust metadata, governance, hardening, and evidence. Version numbers establish repository history only; they are not external certifications.

III. LANGUAGE AND COMPILER ARCHITECTURE

ArgorixLang programs group declarations into modules. Construct families include agents and capabilities, typed messages and handlers, tools and model calls, policies, provider contracts, passports, identity and credential metadata, handshakes, trust ledgers, evidence maps, governance profiles, threat models, runtime profiles, and release/conformance records. Some constructs alter runtime behavior; others are validated descriptions whose execution is explicitly disabled.

Table IV shows why "represented in the language" is not one claim: typed agents and runtime policy have implemented behavior, whereas ATrust evidence maps and provider contracts remain validated descriptions at the boundary discussed here.

The pipeline has five relevant stages. Parsing produces a source-faithful abstract syntax tree. Name and module resolution bind declarations without discarding provenance. Semantic analysis checks uniqueness, references, cross-construct constraints, and denied combinations. Lowering converts only validated forms to intermediate representation and serializable bytecode. Finally, the VM loads bytecode, installs explicitly executable providers, and evaluates policy before scheduling work.

This order matters. A runtime check alone discovers invalid configuration after deployment; an unchecked lowering pass can erase the distinction between an omitted field and an approved field. Semantic validation rejects, among other cases, unknown references, duplicated names, external execution without required approval, inconsistent provider/capability bindings, permissive runtime profiles, and trust declarations that assert prohibited security claims. The conformance corpus contains positive and negative examples for these boundaries.

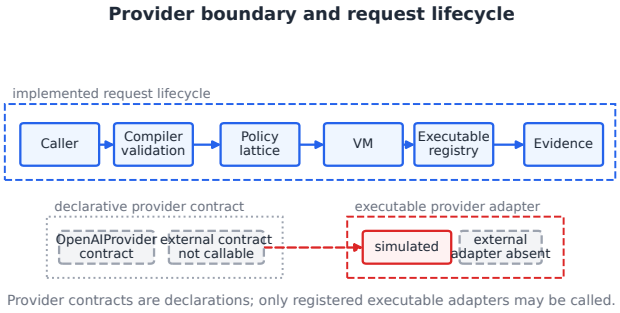

Fig. 2. Provider contracts are declarations; only registered executable adapters may be called. In the evaluated path, simulated is the only executable provider. The diagram does not imply live external-provider execution or production sandboxing.

Figure 2 answers RQ2 at the architectural level. A request cannot directly select an arbitrary provider. The VM resolves the agent and handler, evaluates policy, checks capabilities, and consults the executable provider registry. Declarative contracts such as `OpenAIProvider` remain descriptions unless an independently executable adapter is registered. In the observed configuration, the registry contains simulated; attempts to cross the external boundary are recorded as blocked.

The language also preserves negative statements. Fields such as network_allowed=false, denied key material, mandatory audit, and fail-closed policy are not comments; their

combinations are checked. However, the presence of these fields proves only that the compiler accepted the declaration and the runtime recorded its configured handling. It does not prove operating-system containment.

IV. Sovereign Identity Metadata

An Agent Passport collects locally asserted identity and governance context: passport name, subject agent, country, jurisdiction, permitted data residency, capabilities, policy bindings, and an ans_name. Semantic validation ensures that references resolve and that enumerated relationships are well-formed. Runtime traces can therefore identify which passport and residency constraints accompanied an operation.

The term *sovereign* refers here to explicit jurisdictional and residency metadata controlled by the program, not state recognition or legal status. The observed complete sessions each contain two passports, jurisdiction CL, country CL, and residency values CL and EU. These values are declarations. No external registry validates them.

ans_name is implemented as a declarative passport identifier. It survives parsing, semantic validation, lowering, and artifact generation. It is not an operational Agent Naming Service: no DNS query, DID resolution, registry lookup, ownership challenge, or remote authentication occurs. The implemented relation is local: the name is bound to a passport and agent within the compiled program.

This separation avoids a common category error. A resolvable name answers "where is the record?"; an authenticated identity answers "who controls the relevant key or account?"; a credential answers a scoped assertion; and a policy decision answers whether an operation is permitted in context. A future architecture may connect these functions, but one cannot substitute for another.

The operational cut is immediately after local compilation: federation, DID resolution, and remote proof of control are future components and cannot be inferred from a validated ans_name.

V. ATrust and DCP-AI

ATrust is attributed to Edison Vazquez as an unpublished conceptual protocol [24]. In the conceptual model, an agent identity relates to credential descriptions, handshake requirements, trust-ledger entries, and an evidence map. ArgorixLang implements syntax, semantic relationships, bytecode representations, and trace events for these concepts. It does not implement real credential issuance or verification, cryptographic challenge–response, remote DID resolution, or live attestation.

Identity declarations associate agents with identifier methods and boundaries. Credential contracts describe issuer/subject/type relationships. Handshake declarations describe participants and required evidence. A trust ledger is a local, hash-linked declarative structure whose entries reference subjects and evidence. Evidence maps require named declarations to be covered by artifact links. Validation rejects dangling references and prohibited claims. The word *handshake* therefore denotes a checked contract, not an executed cryptographic exchange; *ledger* denotes a local evidence data structure, not blockchain consensus or immutable storage.

TABLE II
Runtime controls observed in the evaluated profile. These controls describe application-level behavior, not operating-system sandbox proof.

Control	Observed behavior	Scope
Provider execution	simulated only	Implemented
External execution	Blocked	Implemented
Network access	Denied by runtime profiles	Implemented
Secrets and key material	Denied	Implemented

DCP-AI, attributed to Danilo Naranjo Emparanza, proposes a Digital Citizenship Protocol for Autonomous AI and a control-plane framing for agents [25]. It complements ATrust's trust relationships and ArgorixLang's compiled enforcement boundary, but the three are not conflated. DCP-AI is a published conceptual proposal; ATrust is unpublished conceptual work; ArgorixLang is the software artifact evaluated here.

The relationship can be stated operationally. DCP-AI motivates questions of agent standing and governance; ATrust supplies a vocabulary for identity, credentials, handshakes, ledgers, and evidence; ArgorixLang checks local declarations and emits evidence about their handling. None of these local checks demonstrates that a remote party is genuine.

The evidence scope figure in figure 4 distinguishes the unverified source input from bundle-verified JSON artifacts. Its scope does not extend to an issuer, credential authority, or remote agent.

MCP and A2A bridges follow the same principle. They bind agents, passports, identities, message contracts, and boundaries while requiring network and execution denial in the studied profile. The declarations can support future protocol adapters [2, 3]; today they are declarative boundaries and auditable negative controls.

VI. Governed, Fail-Closed Runtime

The runtime evaluates a validated program rather than accepting unrestricted provider calls. Its decision inputs include the requesting agent, capability, target, policy, runtime profile, and executable-provider registry. Denial is the default when a reference is unknown, an approval is absent, a capability is incompatible, or a provider is not executable. Runtime state is retained in an outcome object even when execution fails, allowing a report to describe the denial without converting it into success.

The provider boundary has two registries. Declarative contracts state capabilities, limits, and allowed operations for an external provider. Executable providers contain callable implementations. A contract's presence does not enroll it for execution. The evaluated chatbot artifacts name an OpenAI Provider contract and an OpenAISandbox descriptor, but the only executable provider is simulated. Each complete trace records one blocked external-provider attempt.

Table II records the four controls actually observed, while leaving operating-system containment and live-provider behavior outside the table's scope.

Runtime profiles require audit, evidence, security reports, sandbox metadata, and deny network, plaintext secrets, key material, tools, agent execution, and external execution for the demonstrated path. Governance and threat-model declarations are likewise runtime-disabled. These choices make the corpus suitable for studying compilation and evidence without contacting an external service.

Figure 2 also clarifies the meaning of *fail closed*. It is a control-flow property of the implemented VM: missing authorization does not fall through to a provider call. It is not a claim that every host-language, dependency, operating-system, or deployment path is isolated. A production assessment would need process, filesystem, network, secret-store, and adapter testing beyond the present study.

Policies can activate actions, emit violations, and require human review. Coarse status fields are not authoritative when a detailed policy object is available. This rule becomes consequential in section IX: the corpus contains a favorable top-level label alongside uniformly unfavorable detailed policy results.

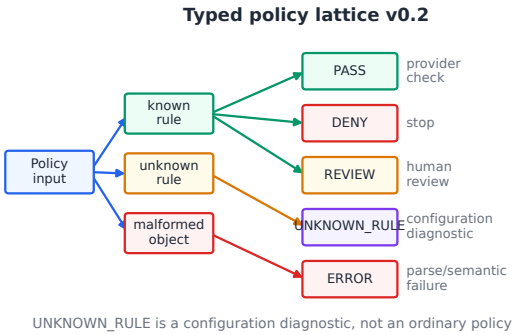

Fig. 3. The v0.2 lattice separates policy approval, denial, review, configuration errors, and malformed policies. UNKNOWN_RULE is not treated as an ordinary violation, and the figure does not imply legal or operational approval outside the local controlled matrix.

VII. Trace, Report, and Evidence Model

Each complete request directory contains five artifacts: session.argx, compiled session.argbc.json, session.trace.json, session.security.json, and session.evidence.json. Source is an upstream compiler input, not an EvidenceBundle member. The trace is an ordered ledger of compiler/VM and policy events. The SecurityReport is a structured interpretation of runtime evidence, including execution status, policy results, review state, provider boundary, and declared controls. The EvidenceBundle records four semantic SHA-256 values: bytecode_digest, trace_digest, report_digest for the security report, and ledger_digest, which is the canonical digest of trace.events. Its artifact map contains bytecode_path, trace_path, and security_report_path; there is no source path or source digest, and there is no standalone ledger path.

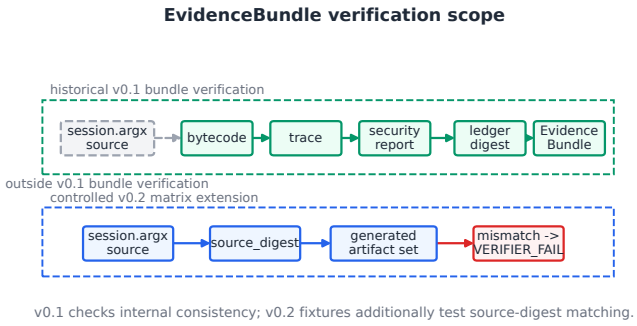

Fig. 4. Offline evidence scope for historical v0.1 bundles and source-digest extension in the controlled v0.2 matrix. v0.1 verifies internal consistency among bytecode, trace, report, and ledger digest; v0.2 controlled fixtures additionally test source-digest matching and source-mismatch detection. The figure does not establish authorship, trusted time, external attestation, or absence of side effects.

A semantic digest is computed over canonical JSON derived from a deserialized value, avoiding dependence on whitespace or object-key formatting. The three artifact paths resolve relative to the bundle directory. Offline verification loads bytecode, trace, and the security report and recomputes their three semantic digests. It separately computes ledger_digest from trace.events. Finally, it checks that the ledger_digest stored in the SecurityReport's LedgerSummary equals the bundle's ledger_digest; the report stores this summary digest, not ledger events. It requires neither network access nor provider credentials. This verifies the relationships encoded by the bundle. Source integrity is outside EvidenceBundle verification.

Figure 4 exposes this asymmetry: a complete request directory contains source, but the bundle verifier starts at compiled bytecode and never asserts that session.argx produced it.

Three distinctions are essential. First, a valid digest detects content change relative to the bundle but does not prove who created the bundle. Second, a linked trace does not prove that events outside the instrumented runtime did not occur. Third, artifact integrity does not turn a failing policy result into approval. No signature, trusted timestamp, transparency service, hardware root, or independent witness is evaluated here.

The local trust ledger is similarly bounded. Hash-linked entries and validated references can expose accidental or adversarial modification when the verifier has an expected root or bundle. They do not provide distributed consensus, blockchain immutability, or non-repudiation. The declarations explicitly deny such security claims.

Figure 5 is the interpretive key for the remainder of the paper: each result is assigned to an implemented, declarative, proposed, or not-claimed region before it is discussed.

Table III applies the same distinction to concrete system concepts, preventing digest success from being promoted to live attestation or certification.

TABLE III
INTERPRETIVE BOUNDARIES APPLIED TO EVERY RESULT IN THIS STUDY.

Concept	Implemented	Declarative	Proposed	Not claimed
Provider boundary	simulated allowlist	contract metadata	–	external execution
Trust evidence	digest bundle	ATrust map	–	live attestation
Discovery	local ans_name binding/metadata	sovereign metadata	federation	operational DNS
Security scope	fail-closed controls	threat mappings	deployment study	certification

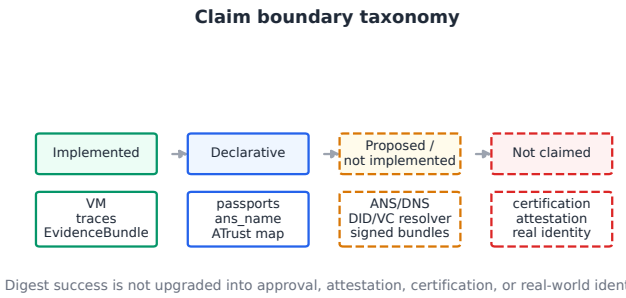

Fig. 5. Claim-boundary taxonomy for interpreting every result. It separates implemented mechanisms, declarative metadata, proposed/not implemented architecture, and properties not claimed; digest success is not upgraded into approval, attestation, certification, or real-world identity.

TABLE IV
REPRESENTATIVE CONSTRUCT FAMILIES AND THEIR BOUNDARY IN THE STUDIED IMPLEMENTATION.

Construct family	Representation	Boundary
Agents and messages	Typed declarations	Implemented
Provider contracts	Declarative metadata	Declarative
Policies and passports	Semantic plus runtime controls	Implemented
ATrust evidence maps	Validated evidence links	Declarative

VIII. METHODOLOGY AND SYSTEM DESIGN

A. System boundary

We evaluate ArgorixLang as a systems artifact for compiled governance, not as a security certification. The implemented path consists of language constructs, semantic validation, typed IR/bytecode, a fail-closed VM, an executable-provider registry, trace generation, a SecurityReport, and an EvidenceBundle verifier. The declarative path consists of Agent Passport fields, ATrust and DCP-AI metadata, external-provider contracts, bridge declarations, and local ans_name metadata. Proposed paths include source-digest binding, signed bundles, append-only logs, DID/VC resolution, and Sovereign DNS/ANS federation.

Table IV is a method table, not a marketing claim: represented constructs are classified by what the current artifact actually does with them.

TABLE V
TYPED POLICY DECISION LATTICE USED IN THE CONTROLLED V0.2 MATRIX.

Outcome	Meaning	Current status
PASS	Known rule satisfied	Implemented in controlled matrix
DENY	Known rule violated and execution must stop	Implemented in controlled matrix
REVIEW	Known rule requires human review	Implemented in controlled matrix
UNKNOWN_RULE	Unsupported policy identifier or configuration error	Implemented as diagnostic outcome
ERROR	Malformed policy object or parse/semantic failure	Implemented in controlled matrix

B. Policy decision lattice

The v0.1 snapshot exposes a design defect: all detailed policy findings are reported as "unknown policy rule." We therefore do not treat those findings as ordinary deny/pass observations. Instead, we specify a v0.2 typed policy decision lattice that future artifacts must implement before policy outcomes can be analyzed as controlled results.

Table V separates known pass/deny/review outcomes from configuration errors and malformed policy objects. Controlled-matrix known rules are visualized in figure 3 and include prov ider_executable_allowlist, external_provider_without_adapter, runtime_network_denied, secrets_denied, key_material_d enied, passport_residency_declared, passport_residency_c onflict, source_digest_present, bundle_digest_match, and evidence_report_consistency. The observational v0.1 snapshot remains reported as a negative diagnostic finding, while the controlled v0.2 matrix generates the typed outcomes used for the policy-lattice evaluation.

C. Snapshot inventory and normalization

The observational input is an authorized snapshot of 33 request directories from the chatbot runtime's generated output. Publication authorization does not relax minimization: identifiers are aggregated, excerpts are bounded, and prompt content is not reconstructed. A complete directory must have all five expected artifacts listed in section VII; 27 meet that criterion and six contain source only. We retain incomplete directories in the denominator and represent unavailable values as missing.

TABLE VI
COMMITTED NORMALIZED DATASETS. VALUES ARE REGENERATED FROM THE AUTHORIZED ARTIFACT SNAPSHOT.

Normalized source	Rows / records	Role
runtime_summary.json	33	Session-level normalized summary
sessions.csv	33	Tabular session observations
event_counts.csv	1890	Per-session event-kind counts
verification-results.json	27	Verification result groups

Table VI separates session-level normalization, event-level counts, and verifier outcomes so that totals can be recomputed without treating one row type as another. The analysis parses available JSON, records parse errors, counts sizes and artifact presence, extracts execution and policy fields, counts ledger-event kinds, and records passport, runtime-profile, adapter, provider-boundary, and digest metadata. A separate PowerShell procedure invokes the repository's offline evidence verifier for every complete bundle and stores exit code and bounded standard output.

D. Controlled artifact matrix

The snapshot alone is not a behavioral corpus. We therefore add a controlled artifact matrix for v0.2. Country and residency fields are evaluated as local metadata. The controlled matrix covers 31 national country codes plus EU as an optional regional residency zone. These cases test parser validation, policy typing, residency-conflict classification, and evidence generation. They do not establish legal compliance or physical storage location.

The 31-country matrix evaluates syntactic and semantic handling of jurisdictional metadata. It does not evaluate legal compliance, physical residency, or external identity verification. Cases include positive controls, negative controls, review controls, tamper controls, and unknown/error controls. Each case reports expected outcome, observed outcome, policy decision, provider decision, evidence status, source digest status, trace event count, and whether side-effect execution occurred.

Table VII now separates the observational runtime snapshot from the generated v0.2 controlled matrix. In particular, source-digest matching and source-mismatch detection are generated by the matrix fixtures; the historical runtime EvidenceBundles still do not verify session.argx. The table entries are Generated in controlled matrix cases rather than inferred from the repeated runtime snapshot.

Table VIII reports only metadata coverage over recognized country codes. It is not a statement about legal coverage, datacenter location, or external citizenship verification.

Table IX is the compact paper-ready view of the generated case table; the full row-level CSV remains the auditable source.

TABLE VII
CONTROLLED EVALUATION MATRIX GENERATED FOR V0.2. ROWS DESCRIBE LOCAL METADATA, POLICY, PROVIDER, EVIDENCE, AND SOURCE-DIGEST CASES, NOT LEGAL OR EXTERNAL IDENTITY ASSURANCE.

Case	Expected outcome	Claim status
31 same-country passport cases	PASS; no external side effect	Generated in controlled matrix
six cross-border residency cases	PASS / REVIEW / DENY by profile	Generated in controlled matrix
valid EvidenceBundle	offline verifier passes	Observed for current bundles
valid source digest match	source digest matches fixture source	Generated in controlled matrix
external provider without executable adapter	DENY	Observed as blocked boundary
network attempt under denied profile	DENY	Observed as denied event
plaintext secret or key material attempt	DENY	Observed as denied class
residency mismatch or unknown jurisdiction	REVIEW	Generated in controlled matrix
modified trace / report / bytecode / ledger	VERIFIER_FAIL	Generated in controlled matrix
source mismatch	VERIFIER_FAIL	Generated in controlled matrix
unknown policy rule	UNKNOWN_RULE configuration error	Generated in controlled matrix
malformed policy object	ERROR parse/semantic failure	Generated in controlled matrix

TABLE VIII
PASSPORT JURISDICTION COVERAGE IN THE CONTROLLED MATRIX. EU IS NOT COUNTED AS A COUNTRY; IT IS USED ONLY AS A REGIONAL RESIDENCY ZONE.

Region group	Countries	Valid PASS cases
home	1	1
latin_america	6	6
north_america	2	2
europe	10	10
asia_pacific	7	7
africa_middle_east	5	5

E. Ablations and baselines

The ablation plan follows a systems-paper discipline: remove one boundary at a time and measure false allows, false denies, correct denies, correct reviews, tamper detection rate, verification latency, evidence size, trace overhead, compile time, and policy evaluation time. These measurements are not present in the current snapshot and must not be claimed until generated.

Table X makes the strongest future claims falsifiable: if removing a boundary does not change a target metric, the paper should not treat that boundary as empirically justified.

Table XI keeps comparisons fair: ArgorixLang is scoped to language, policy, provider boundary, and local evidence linkage, while provenance and authorization systems solve different parts of the assurance stack.

F. Analysis rules

We define four interpretation rules in advance. A missing artifact is "unavailable," never "passed." A detailed policy

TABLE IX
CONTROLLED ARTIFACT MATRIX SUMMARY DERIVED FROM RESULTS/CONTROLLED_MATRIX.JSON.

Measure	Value	Interpretation
Total controlled rows	57	Generated deterministic fixtures
Countries tested	31	National ISO alpha-2 metadata only
Valid same-country passport cases	31	One PASS fixture per country
False allow / deny / review	0 / 0 / 0	Outcome-classification checks
Evidence pass / fail	53 / 4	Bundle or digest-consistency status

TABLE X
REQUIRED ABLATION STUDY. METRICS IN THE LAST COLUMN ARE MEASUREMENT OBLIGATIONS, NOT CURRENT RESULTS.

Variant	Removed boundary	Do not claim until measured
Full ArgorixLang v0.2	none	false allow, false deny, review accuracy, tamper rate, latency
w/o typed policy lattice	typed outcomes	unknown-rule collapse and aggregation errors
w/o executable-provider allowlist	adapter gate	external-provider false allows
w/o source digest	source binding	source mismatch detection
w/o ledger digest	trace-event ledger binding	ledger tamper detection
w/o report cross-check	report-to-bundle linkage	summary/detail inconsistency detection
w/o fail-closed default	default denial	missing-authorization false allows
JSON audit logs only baseline	cross-artifact verification	tamper and linkage gaps

result overrides a coarser summary label for policy interpretation. Successful evidence verification means digest and reference consistency only. Declarative records are not counted as externally performed actions.

We aggregate complete-session policy findings and ledger events by summing the normalized per-session fields. We check all four evidence digest values for the sha256: prefix and 64 hexadecimal digits. The verification result is successful only when the verifier exit code is zero and its explicit verified flag is true.

G. Prompt analysis boundary

Although the user authorized publication, we inspect only the bounded trace.injected.content schema field. It is absent from all 27 valid complete traces (0 of 27 present). The six source-only directories contain no trace and are not assessable. Prompt publication is a separate allowlisted operation; its suppression is not evidence of content absence. We therefore perform no prompt-level qualitative analysis, quote no prompts, and report no prompt-injection attack success rate.

TABLE XI
BASELINE COMPARISON BY SCOPE. BASELINES ARE NOT ADVERSARIES; THEY IDENTIFY WHICH ASSURANCE LAYER EACH SYSTEM DOES OR DOES NOT COVER.

Baseline	What it covers	Gap relative to ArgorixLang
Plain application logs	chronological observations	no compiled policy or digest bundle
JSON audit report	structured summary	no cross-artifact verification
OPA/Rego-style external policy	policy decision engine	no compiled passport/evidence constructs unless integrated
in-toto-style provenance	signed supply-chain steps	different scope; Argorix currently unsigned local runtime evidence
MCP/A2A declaration only	protocol surface description	no compiled fail-closed provider boundary

IX. RESULTS

A. Snapshot completeness

Of 33 request directories, 27 (81.8%) contain all expected artifacts and six (18.2%) contain source only. All 27 complete sessions report execution status completed; execution status is unavailable for the six incomplete sessions. The distinction matters: the study cannot determine whether those six requests failed during compilation, were interrupted, or were retained before downstream artifacts were written.

Table XII keeps the six source-only directories in view and reports aggregate totals only where normalized values exist; this prevents complete-case counts from becoming a 33-session success claim.

B. Deterministic artifact production, limited behavioral diversity

The 27 complete traces contain 7,155 counted ledger events, exactly 265 per complete session. Thus the repeated snapshot shows deterministic artifact production but limited behavioral diversity. It does not show how the system responds to varied prompts, providers, jurisdictions, policy sets, or attacks. The event total is useful for reproducibility checks and evidence-size accounting; it is not evidence of scale in the empirical-security sense.

Each complete trace includes the following denial events: ExternalProviderExecutionBlocked, RuntimeNetworkDenied, and RuntimeSecretsDenied. It also records declaration events for agents, policies, passports, governance, threat, release, ATrust, MCP, and A2A metadata. These observations support the implemented provider-boundary and runtime-denial claims in table II; they do not prove operating-system containment.

C. Policy diagnostics

Each of the 27 complete sessions has security_checks= passed at the top level. Nevertheless, each detailed policy object has policy_passed=false, review_required=true, and

TABLE XII
EMPIRICAL TOTALS DERIVED FROM THE NORMALIZED SNAPSHOT. THESE TOTALS ARE DESCRIPTIVE, NOT BROAD WORKLOAD EVIDENCE.

Measure	Derived value	Source
Complete sessions	27	runtime_summary.json
Incomplete sessions	6	runtime_summary.json
Normalized sessions	33	sessions.csv
Complete-session policy violations	1188	sessions.csv
Counted ledger events	7155	event_counts.csv
Prompt-content traces inspected / present	27 / 0	runtime_summary.json
Verified evidence bundles	27 / 27	verification-results.json

44 findings whose reason is "unknown policy rule." The total is therefore 1,188 detailed policy findings.

We treat this as a diagnostic failure of the v0.1 policy-reporting model, not as a normal policy-deny experiment. The correct paper claim is that the current evidence preserves a negative policy state and exposes an aggregation inconsistency. The controlled v0.2 matrix uses the typed policy decision lattice in table V, where unknown rules become configuration errors and known rules can produce pass, deny, or review outcomes.

D. Offline bundle verification

All 27 complete bundles pass the repository's offline verifier (27/27). Their bytecode, trace, security-report, and trace-event-ledger digest fields are syntactically valid SHA-256 identifiers. The successful result answers RQ3 narrowly: given each bundle and its relative artifact paths, the verifier can reproduce the semantic digests, recompute the ledger digest from trace events, check the SecurityReport LedgerSummary's stored digest against it, and validate cross-artifact linkage without network access. The verifier does not digest or verify `session.argx`; Source integrity is outside EvidenceBundle verification.

It does not answer whether policy approved the request; the detailed policy result is negative. It does not authenticate the producer, prove absence of side effects, establish trusted time, or establish semantic correctness of every event. Integrity and approval are orthogonal axes.

E. Controlled v0.2 matrix

The generated controlled matrix contains 57 rows: 31 valid same-country passport fixtures, six cross-border residency fixtures, invalid and unknown jurisdiction fixtures, five provider-boundary fixtures, five tamper/source fixtures, and two unknown/error policy fixtures. All 31 national country codes produce same-country PASS outcomes. Unknown syntactically valid country codes produce REVIEW, malformed country values produce ERROR, and missing residency with a valid country produces REVIEW. The matrix also records EU only as a regional residency zone, not as a country.

Figure 6 and table XIII show that PASS, DENY, REVIEW, UNKNOWN_RULE, ERROR, and VERIFIER_FAIL outcomes are all represented as typed results. The counts are generated from deterministic fixtures and should not be interpreted as prevalence estimates.

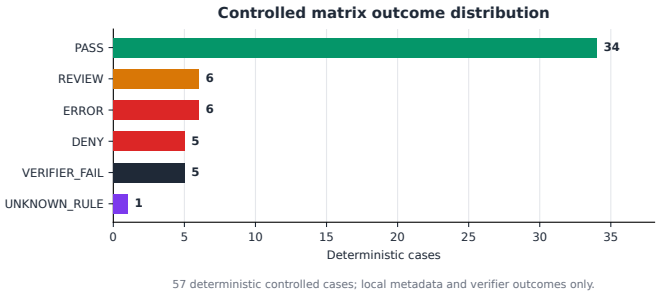

Fig. 6. Controlled v0.2 matrix outcome distribution across 57 deterministic cases. The matrix exercises local metadata validation, policy typing, provider denial, source-digest checks, and tamper detection; it does not evaluate legal compliance, physical residency, or broad workload prevalence.

Table XIV reports verifier failures for modified trace, report, bytecode, ledger events, and source mismatch. Source mismatch is checked in the controlled v0.2 fixtures; it does not retroactively add source checking to the historical runtime EvidenceBundles.

Table XV shows the provider-boundary controls: simulated provider execution is allowed, while external provider without an executable adapter, network attempts, plaintext secret attempts, and key-material attempts are denied with no side effect. The external-provider false-allow count is zero in this controlled matrix.

F. Passports and prompts

Every complete session reports two passports, Chile as country and jurisdiction, and Chilean and European Union residency declarations. These values show that sovereign metadata survives to runtime evidence. They do not show that data was physically stored in either region or that an external authority authenticated the passport.

The bounded `injected.content` field is present in 0 of the 27 complete, valid traces. The six source-only directories have no trace and are not assessable. This presence metric is independent of the publication allowlist: prompt_text remains null unless explicitly allowlisted even when content is present. In this snapshot no qualitative prompt themes, quotations, injection labels, or attack success rates can be reported.

X. SECURITY ANALYSIS

We analyze threats as attack paths across trust boundaries, not as proof that a named control defeats every adversary. The

TABLE XIII
OBSERVED TYPED POLICY OUTCOMES IN THE CONTROLLED MATRIX.

Outcome	Count	Scope
PASS	34	Controlled matrix
DENY	5	Controlled matrix
REVIEW	6	Controlled matrix
UNKNOWN_RULE	1	Controlled matrix
ERROR	6	Controlled matrix
VERIFIER_FAIL	5	Controlled matrix

TABLE XIV
TAMPER AND SOURCE-MISMATCH CONTROLS IN THE CONTROLLED MATRIX.

Case	Observed	Evidence	Source
source mismatch	VERIFIER_FAIL	bundle verified	source digest mismatch
tampered bytecode	VERIFIER_FAIL	bytecode digest mismatch	source digest match
tampered ledger events	VERIFIER_FAIL	ledger digest mismatch	source digest match
tampered report	VERIFIER_FAIL	report digest mismatch	source digest match
tampered trace	VERIFIER_FAIL	trace digest mismatch	source digest match

TABLE XV
PROVIDER-BOUNDARY CONTROLS IN THE CONTROLLED MATRIX.

Case	Observed	Provider	Side effect
external provider without adapter	DENY	external blocked no adapter	no
key material attempt	DENY	no provider planned	no
network denied profile	DENY	no network call	no
plaintext secret attempt	DENY	no provider planned	no
valid simulated provider	PASS	simulated allowed	no

relevant assets are authorization state, source and bytecode, policy results, provider selection, credentials and secret material, trace/report integrity, and jurisdictional metadata. Adversaries may supply malformed programs, attempt unauthorized operations, tamper with stored artifacts, exploit a provider adapter, or induce a consumer to accept a favorable summary while ignoring detailed failures.

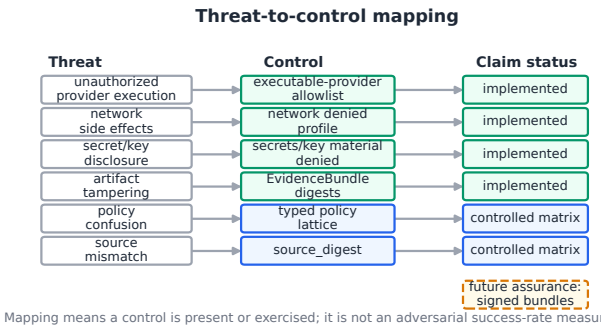

Fig. 7. Threat-to-control mapping for the evaluated architecture and controlled matrix. Mapping means a control is present or exercised in the controlled matrix; it is not an adversarial success-rate measurement, security certification, or proof of production isolation.

Figure 7 visualizes which boundaries a threat crosses, and table XVI states the narrower mitigation claim supported at each boundary; neither reports measured adversarial efficacy.

TABLE XVI
PRINCIPAL THREATS AND BOUNDED MITIGATION CLAIMS.

Threat	Mitigation	Claim status
Unauthorized provider execution	Executable-provider allowlist	Implemented
Network side effects	Offline and network-denied profiles	Implemented
Secret disclosure	Secret and key-material denial	Implemented
Evidence tampering	Digest-linked artifact bundle	Implemented

a) Unauthorized provider execution.: The split between declarative and executable registries prevents a provider contract from becoming callable merely by being named. Capability checks, approval requirements, and a simulated-only allowlist reduce the observed attack surface. A malicious executable adapter or VM defect remains in scope for future testing.

b) Network and secret side effects.: The demonstrated runtime profile denies network access, external execution, plaintext secrets, and key material. Events make these decisions visible. Because this study does not instrument the host operating system, the evidence supports VM-level denial behavior, not production-grade isolation.

c) Artifact tampering.: Semantic digests detect changes relative to an EvidenceBundle and avoid formatting-dependent

hashes. An attacker able to replace both artifacts and the unsigned bundle can create a self-consistent replacement. Signatures, trusted timestamps, independent publication, and key governance are outside the implementation evaluated here.

d) Policy confusion.: The observed status contradiction is an integrity-of-interpretation threat. A dashboard reading only security_checks could present an unsafe conclusion even when the detailed object requires review. Consumers should apply a precedence rule: any explicit failure, unknown rule, or required review prevents an aggregate pass. Future schemas should compute one canonical tri-state decision from typed component results.

e) Prompt injection and content attacks.: Prompt injection is a recognized application risk [26]. ArgorixLang can constrain resulting operations through capabilities and policy, but this corpus contains no prompt text and no controlled attacks. We make no claim of universal injection prevention and report no success rate.

f) Identity and trust spoofing.: Local semantic checks prevent dangling identity and credential references. They do not establish control of a DID, truth of a credential, or execution of a cryptographic handshake. A future resolver and verifier would introduce new trust anchors, replay concerns, revocation, privacy, and network-failure paths [6, 7].

The strongest supported security statement is therefore conditional: within the observed runtime path, requests are mediated by compiled policy and an executable-provider allowlist, denials are recorded, and complete artifacts are offline-consistent. Security outside that path remains unmeasured.

XI. Discussion

a) RQ1: governance as compilation.: The construct inventory shows that identity, residency, provider, policy, governance, threat, and evidence relationships can be represented in one language and rejected before execution when internally inconsistent. The benefit is not that declarations become true; it is that assumptions become typed, referentially checked, and available to both runtime and evidence.

b) RQ2: constrained request planning.: The compiler/VM pipeline and provider split place validation, policy, capability, and registry checks before planning. The observed traces contain blocked external attempts and simulated-only execution. The result supports a fail-closed application architecture, while leaving host isolation untested.

c) RQ3: evidence and offline consistency.: Trace, SecurityReport, and EvidenceBundle serve different functions: event history, interpreted runtime state, and content linkage among bytecode, trace, security report, and the ledger digest derived from trace events. The report's LedgerSummary stores that digest, not the events. All complete bundles verify offline; the upstream source is not part of that verification. The policy contradiction proves why these artifacts should not be collapsed: valid evidence can faithfully preserve an unfavorable result.

d) RQ4: demonstrated and undemonstrated behavior.: The snapshot demonstrates repeatable artifact production for 27 complete sessions, six visible incomplete sessions, stable control events, and offline digest verification. It does not demonstrate prompt robustness, diverse workloads, live providers, identity authentication, credential verification, cryptographic handshakes, production containment, or compliance.

e) RQ5: open agentic-web placement.: Project NANDA provides a research direction for indexed, verifiable agent facts [4, 5]; DCP-AI frames digital citizenship and governance [25]; ATrust supplies an unpublished conceptual trust vocabulary [24]. ArgorixLang can compile local metadata that might feed such systems, but it is not their resolver, registry, or trust authority.

This positioning differs from protocol conformance. MCP and A2A define interaction protocols [2, 3]; ArgorixLang presently checks declarative bridge boundaries with execution disabled. Similarly, DID and VC standards specify interoperable representations [6, 7]; the language stores descriptions without resolving or verifying them.

The policy contradiction is also a design lesson for evidence-consuming systems. "Passed" should not be a free string replicated at multiple levels. A typed lattice such as *deny* $>$ *review* $>$ *allow*, with unknown treated as review or deny, would make aggregation monotone. An EvidenceBundle should then bind both component results and the derived decision, enabling a verifier to recompute the aggregate.

Finally, the repeated 265-event trace suggests that exhaustive metadata can become noisy. Evidence maps and report summaries can improve navigation, but summary must remain traceable to component events. Compact evidence is useful only if it cannot erase negative findings.

XII. Limitations and Claim Boundaries

The dataset is small, single-application, and structurally homogeneous. Its 33 directories are not an independent random sample, and the 27 complete traces have identical event totals. Percentages are descriptive only. Six source-only directories have unknown outcomes, introducing survivor bias if analysis is restricted to complete sessions.

The v0.2 controlled matrix is deterministic and local; it is not a randomized workload, legal analysis, external identity test, physical data-residency measurement, adversarial campaign, concurrency study, performance benchmark, or usability study. The bounded prompt-content field was present in 0/27 complete valid traces; six source-only directories were unassessable. Thus prompt-level qualitative analysis and injection measurement are impossible. We do not infer attack success from policy violations: those violations report unknown rules, not successful attacks.

Offline verification checks bytecode, trace, security-report, and trace-event-ledger digest consistency against an unsigned local bundle. It does not check source integrity, authenticate authorship, establish trusted time, prevent replacement of the entire bundle, or prove that the trace is complete. SHA-256 is used for semantic digests; no post-quantum security property is evaluated or claimed.

TABLE XVII
LIMITATIONS AND FUTURE ASSURANCE REQUIREMENTS. THESE ARE EXCLUDED FROM CURRENT CLAIMS UNLESS THE REQUIRED MECHANISM AND EVALUATION ARE ADDED.

Limitation	Current status	Future assurance requirement
Live provider execution	Not implemented	sandboxed adapter plus host telemetry
Real identity authentication	Not implemented	DID/VC or equivalent resolver and issuer policy
Source integrity	Outside current bundle verification	source digest and source-to-bytecode binding
Producer authentication	Not implemented	signed bundles and key governance
Replacement detection	Not implemented	append-only log or transparency service
Large workload diversity	Not present	controlled workload matrix and pre-registered metrics

Passports, ans_name, jurisdictions, residency, ATrust identities, credential contracts, handshakes, ledgers, MCP bridges, and A2A bridges are partly or wholly declarative. The current system does not resolve Sovereign DNS/ANS names or DIDs, authenticate real agents, verify credentials, execute cryptographic handshakes, provide blockchain consensus or immutable storage, or run live MCP/A2A interoperability.

Application-level deny flags and the provider registry do not establish production isolation. We did not audit dependencies, prove memory safety of all unsafe/transitive code, inspect operating-system calls, or test container escapes. We make no guarantee against all prompt injection, data leakage, or malicious providers.

Governance mappings are not compliance certification. Neither the language nor this study certifies conformity with a law, regulator, NIST AI RMF, or any other framework [27]. The empirical contradiction between the top-level and detailed statuses further prevents a positive assurance conclusion.

Table XVII converts limitations into concrete future requirements: source integrity needs source-to-bytecode binding, producer authentication needs signatures and key governance, and replacement detection needs an append-only transparency mechanism.

Future evaluations should publish controlled workloads, independent verifiers, OS-level telemetry, and pre-registered outcome definitions.

XIII. FUTURE WORK: SOVEREIGN DISCOVERY

The proposed Sovereign DNS/Agent Naming Service would resolve an ans_name to a versioned agent record containing endpoints, passport references, supported protocols, jurisdictional metadata, and evidence locations. It would draw on DNS lessons concerning hierarchical delegation [14, 15], NANDA's agent-index direction [5], and DID/VC representations [6, 7]. This is proposed architecture, not implemented resolution.

A credible implementation requires more than a lookup API. It needs namespace governance, proof of control, authenticated record updates, revocation, key rotation, replay protection, privacy-preserving queries, caching semantics, conflict handling, auditability, and resilience to compromised registries. Jurisdiction and residency assertions need issuer and evidence policies rather than self-assertion alone.

ATrust contracts could define which identity methods, credential types, and handshake evidence a caller requires. DCP-AI could inform citizenship and control-plane semantics. ArgorixLang could compile these requirements and reject a discovered record before an adapter is planned. An executable resolver, credential verifier, and cryptographic handshake engine would nevertheless be new trusted components, requiring independent threat models and tests.

Empirically, future work should generate a controlled matrix: valid and invalid records, stale keys, revoked credentials, cross-jurisdiction residency conflicts, unavailable registries, malicious adapters, altered artifacts, and summary/detail contradictions. Outcome measures should include denial correctness, false acceptance, false rejection, verification latency, evidence size, and recovery behavior. Prompt robustness should be studied only with an explicitly collected, privacy-reviewed corpus.

For evidence, signed bundles and append-only transparency publication could provide producer authentication and replacement detection, subject to key governance and availability analysis. This direction should not be described as blockchain immutability unless an actual consensus/storage system is implemented and evaluated.

A. Staged roadmap

We sequence the proposed work as a claim-bounded roadmap, in which each stage adds a single trust property and its own evaluation rather than bundling assurance claims. The version labels are planning milestones, not released guarantees.
v0.1 Compiled governance with offline-verifiable EvidenceBundles (the configuration evaluated here).
v0.2 Include a source digest in the EvidenceBundle so that compiled bytecode can be tied back to its source.
v0.3 Signed bundles and an append-only transparency log for producer authentication and replacement detection.
v0.4 A live provider adapter executed under explicit sandbox constraints, with host-level telemetry.
v0.5 A DID/VC resolver and credential verification, introducing the associated trust anchors and revocation paths.
v1.0 Independent, production-oriented evaluation against controlled workloads and pre-registered outcome definitions.

XIV. CONCLUSION

ArgorixLang demonstrates how agent declarations, policy, sovereign metadata, provider boundaries, and evidence requirements can share a compiled systems boundary. In an authorized observational snapshot, 27 of 33 request directories were complete, six were source-only, and all 27 complete EvidenceBundles verified bytecode, trace, security-report, and trace-event-ledger digest consistency offline; source was outside verification. The same complete sessions contained

1,188 detailed policy violations and uniformly required review despite a favorable top-level status. That contradiction is not incidental: it establishes the paper's central distinction between integrity and approval.

The implemented contribution is a validated language, fail-closed VM path, simulated-only executable-provider boundary, runtime trace, SecurityReport, EvidenceBundle, and semantic digest verification. Passports and ans_name are implemented as declarative identifiers and metadata. ATrust, MCP, and A2A relationships are validated declarations, not live trust or protocol execution. Sovereign DNS/ANS resolution is proposed. Real identity authentication, credential verification, cryptographic handshakes, blockchain immutability, post-quantum security, compliance certification, universal injection prevention, and production isolation are not claimed.

The broader systems lesson is that trustworthy evidence must preserve negative and incomplete outcomes. A verifiable bundle is valuable precisely because it can carry a failed policy decision without relabeling it as success.

Code Availability

The ArgorixLang compiler, runtime, and offline evidence verifier are openly available. The evaluation in this paper corresponds to a single, archived release.

Repository: https://github.com/argorixlabs/argorixlang
Project website: https://www.argorix-lang.org
Documentation: https://www.argorix-lang.org/docs
Evaluated commit: 6f473f5
Release version: v1.0.1
Code license: Apache License 2.0
Archived DOI: https://doi.org/10.5281/zenodo.21007563

The archived DOI resolves to the immutable Zenodo deposit for the release; the repository URL tracks ongoing development and may diverge from the archived state.

Project Website

The project website at https://www.argorix-lang.org hosts the language overview, and the documentation portal at https://www.argorix-lang.org/docs describes language constructs, the runtime profile, and the evidence schema. Website material is informational and does not extend the claims evaluated here; the archived repository and Zenodo deposit remain authoritative for reproduction.

Artifact Availability

The reproducible build pipeline is contained in the paper/ directory of the archived repository. It comprises the analyzer that normalizes the runtime snapshot, the PowerShell driver that invokes the offline evidence verifier, the scripts that render every table and vector figure, the static manuscript checks, and the pinned LaTeX build. The committed normalized datasets, generated tables, and generated figures are derived solely from the authorized runtime snapshot and can be regenerated from it. No executable build tool is committed; the LaTeX engine is downloaded on demand and validated against a published checksum before use. Per-request runtime artifacts referenced throughout the paper are the five-artifact directories described in section VII.

Dataset Availability

The empirical study uses an authorized snapshot of 33 runtime request directories generated by the chatbot runtime. The user authorized publication of this material, but authorization does not relax data minimization. The committed, normalized datasets (session-level, event-level, and verifier-outcome records) contain only aggregated counts and bounded structural fields and may be published in redacted form. Raw prompt text, secrets, key material, provider endpoints, absolute filesystem paths, and other sensitive values are not published. Where the complete corpus is not redistributed, the reason is privacy and data minimization rather than unavailability of results: the normalized records are sufficient to recompute every reported total, while the unredacted inputs remain withheld.

Reproducibility Instructions

The full pipeline normalizes the snapshot, reruns offline bundle verification, regenerates tables and figures, applies static manuscript checks, and renders the preprint. The environment and entry points are as follows; build-host values not fixed by the repository are recorded in the release notes for the archived deposit.

Operating system: developed and validated on Windows; the documented commands target both PowerShell and POSIX shells.
Rust toolchain: as pinned by the repository toolchain configuration.
ArgorixLang version: v1.0.1 (commit 6f473f5).
Python: 3.11 or newer, with the pinned paper/requirements.txt.
Input artifact path: demo/argorix-chatbot-runtime/generated (relative to the repository root).
Output datasets: paper/data, with generated tables and figures in paper/tables and paper/figures.

The canonical entry point regenerates all artifacts:

```
python -m pip install -r paper/requirements.txt
powershell -NoProfile -ExecutionPolicy Bypass \
  -File paper/scripts/build_paper.ps1 -Target all
```

The offline evidence verifier can be rerun in isolation over the snapshot:

```
powershell -File paper/scripts/verify_runtime.ps1 \
  -InputRoot demo/argorix-chatbot-runtime/generated
    \
  -OutputPath paper/data/verification-results.json
```

The expected outcome is deterministic: 27 of 33 complete EvidenceBundles verify (27/27 of the complete set), and the six source-only directories are retained as unavailable rather than discarded. Successful verification confirms offline artifact integrity only; it is not policy approval, a security proof, or certification.

Required Artifact Changes Before Expanded Claims

The paper's stronger v0.2 claims require implementation evidence, not prose alone. Required code and artifact changes are: implement the typed policy decision lattice in table V; add known rules for executable provider allowlists, external adapters, network denial, secret and key-material denial, passport residency, source digest presence, bundle digest matching, and report consistency; add a source digest to the EvidenceBundle and bind it to the compiled bytecode; generate the controlled cases in table VII; implement tamper tests for modified bytecode, trace, security report, and source; record side-effect execution status per case; and publish latency, evidence-size, trace-overhead, compile-time, and policy-evaluation metrics for the ablations in table X. Until those artifacts exist, the current paper must report them as Required experiment, To be generated, or Do not claim until measured.

Reviewer Checklist

A reviewer should be able to answer yes to the following bounded claims and no to the corresponding overclaims. The paper claims a compiled governance language/runtime, a fail-closed provider boundary, and offline internal evidence consistency for 27 complete bundles. It does not claim security certification, real identity verification, cryptographic attestation, live ANS/DNS federation, production sandboxing, broad prompt-injection defense, or a large empirical security corpus. Counts over the repeated snapshot are descriptive. Unknown policy rules are treated as a negative diagnostic finding. Source integrity, signatures, append-only logs, workload diversity, and controlled ablations are future assurance requirements unless their artifacts are generated.

Ethics and Privacy Statement

This study analyzes only material that the user authorized for publication, and it applies data minimization throughout. We do not reconstruct prompts, and we publish no secrets, credentials, endpoints, or otherwise sensitive content. The analysis inspects only the bounded trace.injected.content schema field, which is present in 0 of 27 complete, valid traces; the six source-only directories contain no trace and are not assessable. Because no prompt content is available, we perform no prompt-level qualitative analysis, quote no prompts, and report no prompt-injection success rate. The absence of prompt text reflects a deliberate minimization decision and the studied runtime configuration; it is not evidence of the absence of risk. The work involves no human subjects, personal data collection, or interaction beyond the authorized artifact snapshot.

Author Contributions

The authors jointly contributed to the conception of the system, the language and runtime design, the implementation under evaluation, the observational analysis, and the writing and revision of the manuscript. Detailed CRediT role assignments will be finalized for the camera-ready version.

Funding

No dedicated external grant funded the preparation of this preprint. Any institutional support will be acknowledged in the camera-ready version.

Competing Interests

The authors are affiliated with the organizations developing ArgorixLang and disclose this relationship as a potential competing interest. The authors declare no other competing financial or non-financial interests.

Appendix

The following excerpts are schematic projections of fields present in the normalized data. Values that identify individual requests are omitted; no prompt, secret, endpoint, absolute path, full log, or identifier dump is included.

Listing 1. Redacted session-level projection.

```
{
  "complete": true,
  "artifact_count": 5,
  "execution_status": "completed",
  "security_checks": "passed",
  "policy_passed": false,
  "review_required": true,
  "policy_violation_count": 44,
  "prompt_content_present": false,
  "prompt_text": null
}
```

Listing 2. Redacted provider-boundary projection.

```
{
  "declarative_contract_names": ["OpenAIProvider"],
  "executable_providers": ["simulated"],
  "external_contracts_total": 1,
  "blocked_attempts": 1,
  "external_execution_blocked": true
}
```

Listing 3. Bounded verification-result projection.

```
{
  "exit_code": 0,
  "verified": true,
  "stdout": "Evidence verification: passed",
  "stderr": ""
}
```

From the root of this paper worktree, the authorized artifact snapshot resides in the primary checkout two levels above. The following commands use only root-relative paths and provide every mandatory analyzer argument:

```
python -m pip install -r paper/requirements.txt
powershell -NoProfile -ExecutionPolicy Bypass -File
    paper/scripts/build_paper.ps1 -Target all -
    InputRoot ../../demo/argorix-chatbot-runtime/
    generated
```

In a conventional checkout where generated artifacts are directly under the repository root, replace each ../../demo/ prefix with demo/. The runtime analyzer fails visibly if the input is absent. The explicit -InputRoot prevents a worktree's tracked but empty generated/.gitkeep directory from being mistaken for the authorized snapshot. To rebuild only the PDF with the pinned, checksum-verified Tectonic engine, use:

```
powershell -NoProfile -ExecutionPolicy Bypass -File
    paper/scripts/build_paper.ps1 -Target paper
```

The committed normalized data permit manuscript and figure regeneration without publishing the original request-directory names or prompt content.

REFERENCES

[1] Argorix Labs, "ArgorixLang," 2026, secure, verifiable programming language and runtime for AI-agent systems. [Online]. Available: https://github.com/argorixlabs/argorixlang

[2] Model Context Protocol Contributors, "Model context protocol specification, version 2025-06-18," Protocol specification, 2025. [Online]. Available: https://modelcontextprotocol.io/specification/2025-06-18

[3] A2A Protocol Working Group, "Agent2agent (A2A) protocol specification," Protocol specification, 2026. [Online]. Available: https://a2a-protocol.org/v1.0.0/specification/

[4] Project NANDA, "Architecting the internet of AI agents," Project website, 2026, project NANDA (Networked AI Agents in Decentralized Architecture). [Online]. Available: https://projectnanda.org/

[5] R. Raskar, P. Chari, J. Zinky, M. Lambe, J. J. Grogan, S. Wang, R. Ranjan, R. Singhal, S. Gupta, R. Lincourt, R. Bala, A. Joshi, A. Singh, A. Chopra, D. Stripelis, B. B, S. Kumar, and M. Gorskikh, "Beyond DNS: Unlocking the internet of AI agents via the NANDA index and verified AgentFacts," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2507.14263

[6] World Wide Web Consortium, "Decentralized identifiers (DIDs) v1.0: Core architecture, data model, and representations," World Wide Web Consortium, W3C Recommendation, Jul. 2022. [Online]. Available: https://www.w3.org/TR/2022/REC-did-core-20220719/

[7] ——, "Verifiable credentials data model v2.0," World Wide Web Consortium, W3C Recommendation, May 2025. [Online]. Available: https://www.w3.org/TR/2025/REC-vc-data-model-2.0-20250515/

[8] Open Policy Agent Project, "Open policy agent (OPA)," Official project documentation, 2026, general-purpose policy engine and Rego policy-as-code language. [Online]. Available: https://www.openpolicyagent.org/docs

[9] Cedar Policy Language Project, "Cedar policy language reference guide," Official documentation, 2026. [Online]. Available: https://docs.cedarpolicy.com/

[10] SPIFFE Project, "SPIFFE concepts," Official documentation, 2026. [Online]. Available: https://spiffe.io/docs/latest/spiffe-about/spiffe-concepts/

[11] SPIRE Project, "SPIRE concepts," Official documentation, 2026. [Online]. Available: https://spiffe.io/docs/latest/spire-about/spire-concepts/

[12] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1393–1410. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[13] SLSA Project, "SLSA specification v1.0," Supply-chain security specification, 2023. [Online]. Available: https://slsa.dev/spec/v1.0/

[14] P. Mockapetris, "Domain names—concepts and facilities," Internet Engineering Task Force, RFC 1034, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1034.html

[15] ——, "Domain names—implementation and specification," Internet Engineering Task Force, RFC 1035, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1035.html

[16] T. S. Senarath and A. Dissanayaka, "OAuth 2.0 extension: On-behalf-of user authorization for AI agents," Internet Engineering Task Force, Internet-Draft, 2025, work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-oauth-ai-agents-on-behalf-of-user/01/

[17] FIDO Alliance, "Fido alliance to develop standards for trusted AI agent interactions," Standards working-group announcement, 2026. [Online]. Available: https://fidoalliance.org/fido-alliance-to-develop-standards-for-trusted-ai-agent-interactions/

[18] Agent Payments Protocol Project, "Agent payments protocol (AP2) documentation," Protocol documentation, 2025. [Online]. Available: https://ap2-protocol.org/

[19] Mastercard, "How verifiable intent builds trust in agentic AI commerce," Technical and industry overview, 2026. [Online]. Available: https://www.mastercard.com/us/en/news-and-trends/stories/2026/verifiable-intent.html

[20] OWASP Foundation, "OWASP top 10 non-human identities risks – 2025," Security guidance, 2025. [Online]. Available: https://owasp.org/www-project-non-human-identities-top-10/2025/top-10-2025/

[21] A. Grigor, C. Schroeder de Witt, S. Birnbach, and I. Martinovic, "Vet your agent: Towards host-independent autonomy via verifiable execution traces," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2512.15892

[22] WASI Subgroup, "Introduction," Official standards portal, 2026, standards-track APIs and capability-based

sandbox model. [Online]. Available: https://wasi.dev/
[23] Rust Project, "Rust programming language," Official project website, 2026. [Online]. Available: https://rust-lang.org/
[24] E. Vazquez, "ATrust: Agentic trust protocol," author-supplied metadata used in this study; unpublished conceptual protocol.
[25] D. Naranjo Emparanza, "Agents don't need a better brain—they need a world: Toward a digital citizenship protocol for autonomous AI systems," Engineering Archive preprint, 2026, DCP-AI: Digital Citizenship Protocol for Autonomous AI. [Online]. Available: https://engrxiv.org/preprint/view/6671
[26] OWASP Foundation, "OWASP top 10 for large language model applications," Security guidance, 2025. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/
[27] E. Tabassi, "Artificial intelligence risk management framework (AI RMF 1.0)," National Institute of Standards and Technology, NIST AI 100-1, Jan. 2023. [Online]. Available: https://doi.org/10.6028/NIST.AI.100-1