Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Agentic AI-Based Predictive Security and Resilience Framework for SSL/TLS and Encrypted Traffic Infrastructure in Cloud-Native Environments

Pushpjeet Shrivastava

pushpjeet36@gmail.com

Illinois State University

Vishnu Gatla

San Francisco Bay University

Syam Dondapati

American International Group (AIG)

Srikanth Raju Uppalapati

Cox Enterprises

Research Article

Keywords: Agentic AI, Multi-agent systems, SSL/TLS security, Encrypted traffic analysis, Cloud-native infrastructure, AI guardrails

Posted Date: September 14th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10529066/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

As cloud-native architecture evolves rapidly, certificate lifecycle management, encrypted traffic inspection, and cryptographic standards compliance have become harder to manage in dynamic, heterogeneous environments. Traditional mechanisms do not anticipate cascading failures from certificate expiry, do not detect adversarial traffic within encrypted channels, and do not provide adaptive cipher governance at the required speed and scale. This paper proposes AAPS-TLS, a multi-agent orchestration framework that employs LLM-driven reasoning, reinforcement learning (RL)-based remediation planning, and a custom AI Guardrail Layer to deliver automated, policy-compliant predictive security and resilience for SSL/TLS infrastructure in cloud-native environments. AAPS-TLS comprises five cooperative agentic modules: (i) a Predictive Certificate Intelligence Agent, (ii) an Encrypted Traffic Anomaly Detection Agent, (iii) a Cipher Policy Governance Agent, (iv) a Self-Healing Remediation Agent, and (v) a Compliance and Audit Agent, all coordinated by a central Guardrail Controller that ensures transparency, auditability, and regulatory compliance.

We validate AAPS-TLS primarily through a fully reproducible, open-source discrete-event simulation of cloud-native certificate infrastructure (5,000 TLS traffic flows, 15% malicious, across three cloud regions, with 2,000 managed certificates), for which we report means, standard deviations, and 95% confidence intervals over 30 independent seeded runs and release all code and synthetic data. In this reproducible setting, AAPS-TLS attains near-ceiling certificate-failure prediction (mean 99.8%), encrypted-traffic detection (mean 99.5%), and deterministic policy compliance, while consistently ranking above six baseline approaches. As preliminary, illustrative evidence of real-world applicability, we additionally report descriptive aggregates from a single six-month production-equivalent pilot (1,247 microservices and 1,200 managed certificates across AWS, Azure, and GCP), in which AAPS-TLS reduced certificate-related outages by 91.3%, improved encrypted-threat mean-time-to-detect by 73.6%, and reached 99.2% policy compliance; because this pilot was not replicated and its data is confidential, we treat these figures as indicative rather than as statistically validated claims. The work contributes an open, reproducible framework and design principles for trusted, AI-governed cryptographic-infrastructure management in regulated cloud environments.

1. Introduction

Cloud-native architectures — built on microservices, containerized deployments, service meshes, and dynamic scaling — have become globally prevalent, transforming the threat landscape for SSL/TLS infrastructure. The 2024 Ponemon Institute Cloud Security Report found that 73% of enterprise cloud downtime attributable to cryptographic failures results from cascade events caused by certificates expiring unpredictably, at an average organizational cost of $5.4 million per outage affecting cloud services [1]. The advent of TLS 1.3 and encrypted-by-default communication has rendered standard Deep Packet Inspection (DPI) ineffective; Gartner estimates that more than 70% of malware now uses encryption to evade detection [2].

Current solutions — commercial certificate management platforms (e.g., Venafi, DigiCert CertCentral), SIEM-integrated monitoring, and rule-based policy engines — are reactive, lack cross-domain reasoning, and do not generalize to the combinatorial complexity of cloud-native certificate topologies. As agentic AI — autonomous multi-agent systems that can plan, use tools, retain memory, and self-reflect — becomes feasible, a new mechanism to close this gap is conceivable. Whereas a typical large language model (LLM) simply responds to an input, an agentic system can architect a multi-step workflow from that input, connect to live infrastructure APIs, and adapt its strategy as it observes results.

However, deploying agentic AI in regulated, high-stakes security infrastructure poses major challenges: autonomous decisions must remain explainable, auditable, and enforceable against standards such as NIST SP 800 – 52 Rev. 2 (TLS Guidelines), ISO/IEC 27001, SOC 2, and PCI-DSS. This requires AI Guardrail capabilities — architectural constraints that confine agentic behavior within policy-defined safety envelopes — a dimension little studied in prior security literature on AI.

This paper addresses these challenges through three core contributions:

- A five-agent, guardrail-governed framework (AAPS-TLS) that unifies predictive certificate intelligence, non-decrypting encrypted-traffic anomaly detection, cipher-policy governance, self-healing remediation, and compliance auditing for cloud-native SSL/TLS infrastructure — the first, to our knowledge, to target this domain with cooperating agents rather than isolated point tools. Significance: it moves certificate and traffic security from reactive, per-event handling to dependency-aware, policy-compliant automation.
- The AI Guardrail Layer (AGL), a reusable and reproducible runtime-governance pattern comprising input validation, decision boundaries (Open Policy Agent), output explainability (LIME/SHAP), regulatory alignment, and drift detection (CUSUM), which operationalizes NIST AI RMF and ISO/IEC 42001 as executable software. Significance: it makes autonomous agent behavior auditable and constrainable in regulated environments, a prerequisite for real deployment that prior agentic-security work leaves unaddressed.
- A fully reproducible, open-source evaluation on a seed-deterministic simulation with rigorous statistical reporting (means, standard deviations, and 95% confidence intervals over 30 runs) and released code and synthetic data, complemented by preliminary illustrative evidence from a single production-equivalent pilot. Significance: the primary claims can be independently verified, directly addressing reproducibility concerns common to agentic-AI and security systems research.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the threat model and problem formulation. Section 4 presents the AAPS-TLS architecture. Section 5 details the AI Guardrail Layer. Section 6 covers experimental design and evaluation. Section 7 discusses findings, limitations, and future work. Section 8 concludes.

2. Related Work

2.1 SSL/TLS Certificate Lifecycle Management

The ACME protocol [3], standardized in RFC 8555, laid the groundwork for automatic certificate issuance and renewal; coverage has since been extended incrementally, including RFC 8657 for subdomains and RFC 9115 for certificate revocation. Enterprise lifecycle-management solutions on commercial platforms such as Venafi Trust Protection Platform [4] and AppViewX CERT+ [5] are event-driven and reactive, with architectures that do not address a predictive horizon for elaborate cloud-native dependency graphs. Shrivastava and Kumar [6] show how, in service meshes (Istio, Linkerd) within microservice environments, inter-service TLS dependencies produce cascading failure modes that existing tools cannot predict. Our work extends these findings by providing an agentic reasoning layer that models certificate dependency graphs and predicts failure propagation.

2.2 Encrypted Traffic Analysis

Analyzing malicious traffic within encrypted channels without decryption has attracted substantial research. Anderson and McGrew [7] pioneered the use of TLS metadata (handshake features, packet timing, certificate characteristics) for malware family classification using Random Forests, achieving 93.2% accuracy on the ISCX dataset. Rezaei and Liu [8] applied deep learning to TLS flow classification, while Lotfollahi et al. [9] proposed deep packet inspection via deep learning on raw payloads. More recent surveys map the rapid growth of this area: Shen et al. [30] provide a comprehensive survey of machine-learning-powered encrypted-traffic analysis, and Luxemburk et al. [32] extend classification to the QUIC setting. A recurring and practically important challenge is concept drift: Malekghaini et al. [31] show empirically that deep-learning classifiers for encrypted traffic degrade under data drift over time, motivating the explicit drift-detection guardrail we introduce in Section 5.5. Wang et al. [10] proposed federated learning for distributed encrypted traffic classification, an orthogonal contribution whose privacy-preserving properties complement our architecture. Unlike these works, which focus on the classification task in isolation, AAPS-TLS couples encrypted-traffic detection with autonomous, policy-governed remediation.

2.3 Agentic AI in Cybersecurity

The application of agentic AI to cybersecurity has accelerated with capable LLM foundations. PentestGPT [11] demonstrated LLM-guided penetration testing; AutoAttacker [12] showed end-to-end autonomous red-teaming using GPT-4-class models. In defensive contexts, ReAct-based [13] security orchestration agents have been proposed for SOAR workflows. To our knowledge, no prior work has formalized an agentic AI architecture specifically targeting SSL/TLS infrastructure, nor addressed the AI Guardrail requirements for agentic deployment in production cryptographic-management contexts.

2.4 AI Guardrails and Trustworthy Autonomous Systems

The NIST AI Risk Management Framework (AI RMF) [14] and ISO/IEC 42001:2023 [15] provide high-level guidance for responsible AI deployment. Perez and Ribeiro [16] introduced formal prompt-injection taxonomies for LLM-based systems. Constitutional AI [17] and RLHF [18] offer training-time alignment

approaches but do not address runtime governance in agentic pipelines. Our AI Guardrail Layer (Section 5) operationalizes these frameworks into concrete, reproducible architectural components — a gap in existing literature, where guardrail concepts are rarely instantiated and evaluated as runtime software.

2.5 LLM and Agentic AI for Network and Service Management

AAPS-TLS also relates to the fast-growing body of work applying large language models and agentic AI to network and service management. Hong [33] surveys LLM-based network management and operations, and Leivadeas and Falkner [40] survey the broader intent-based networking paradigm within which many of these systems sit. Concrete systems include LLM-centric intent management for next-generation networks [34], intent extraction for 5G core networks [35], LLM-based policy generation for intent-based application management presented at CNSM [36], in-context configuration for network function virtualization presented at NOMS [37], and zero-touch configuration management [38]. Machine-learning techniques have likewise been applied to network assurance in data centers [39]. These efforts concentrate on configuration, intent translation, and assurance for general network operations; none targets the SSL/TLS and encrypted-traffic security domain, and none couples multi-agent operation with an explicit, reproducible runtime governance layer. AAPS-TLS is, to our knowledge, the first to bring agentic, guardrail-governed management specifically to cryptographic infrastructure, positioning it within — and extending — the network-and-systems-management research line.

3. Threat Model and Problem Formulation

3.1 Threat Model

We consider a cloud-native enterprise environment deploying N microservices $\{s_1, s_2, \ldots, s_N\}$ over K cloud regions, where service s_i maintains one or more TLS certificates $\{c_{i,1}, \ldots, c_{i,M_i}\}$ and communicates over encrypted channels. We categorize the following threat types:

- T1 — Certificate Expiry Cascade: Unmanaged expiry of leaf or intermediate CA certificates that propagates through dependent service chains, causing service unavailability.
- T2 — Weak Cipher Exploitation: Exploitation of deprecated or misconfigured cipher suites (e.g., legacy TLS versions, weak key exchange) to downgrade sessions or enable cryptographic attacks.
- T3 — Encrypted C2 Beaconing: Command-and-control traffic embedded within TLS sessions, leveraging the encrypted channel to evade perimeter detection.
- T4 — Certificate Transparency Abuse: Adversaries leveraging CT logs to identify certificate issuance patterns and time attacks against certificate rotation events.
- T5 — Man-in-the-Middle via Rogue Certificates: Fraudulently issued or compromised intermediate CA certificates that enable impersonation attacks.

- T6 — Agentic AI Prompt Injection: Adversarial inputs embedded in infrastructure telemetry designed to manipulate agentic reasoning and induce policy violations.

3.2 Problem Formulation

Let C = {c_1, …, c_n} denote the set of managed certificates, each with state vector s_i = (expiry_date, chain_depth, cipher_suite, SAN_count, rotation_history, anomaly_score). Let G be the service dependency graph with vertices V = services and edges E = TLS-encrypted inter-service calls. Given G and a time-series history of state vectors H = {s(t − k), …, s(t)}, the predictive-security problem is to learn a policy π: H → A that selects remediation actions A = {renew, rotate_cipher, quarantine, alert, no-op} so as to minimize the expected risk R(t + Δ) over a time horizon Δ, subject to regulatory compliance constraints C_reg (NIST, PCI-DSS, ISO 27001) and budget constraints B:

$\pi^* = \text{argmin}_\pi \, E[R(t + \Delta) \mid H, G]$ subject to C_reg, B

This formulation captures the sequential decision-making nature of SSL/TLS security management, making it suitable for reinforcement learning while encoding compliance rules explicitly.

4. AAPS-TLS Framework Architecture

4.1 Agent 1: Predictive Certificate Intelligence Agent (PCIA)

The PCIA maintains a live knowledge graph of all managed certificates and their inter-service dependencies, predicts failure events, and schedules proactive renewals. It employs a Temporal Graph Neural Network (T-GNN) [20] to model certificate dependency propagation across the service graph G, and an LLM-based Natural Language Interface (NLI) for interpreting certificate-authority advisory feeds, CVE databases, and CT-log anomalies.

The PCIA prediction model is trained on a proprietary dataset of 4.2 million historical certificate lifecycle events across 14 enterprise environments (2019–2024), augmented with synthetic failure scenarios generated via Monte Carlo simulation. Given a certificate c_i with state vector $s_i(t)$, the PCIA outputs a failure probability distribution P(failure | s_i, G, t + Δ) for planning horizons Δ ∈ {7, 30, 90} days, along with a ranked dependency-impact list.

4.2 Agent 2: Encrypted Traffic Anomaly Detection Agent (ETADA)

ETADA examines encrypted traffic using telemetry without decrypting it. It uses eBPF-based kernel probes to extract flow-level features — packet inter-arrival times, burst patterns, and payload-length distributions — enriched with TLS-handshake metadata including certificate fingerprints, JA3/JA3S hashes, and ALPN extensions. Two complementary models operate on these features: a transformer

sequence encoder that captures temporal context across prior flows, and a contrastive-learning model trained on benign TLS behavior.

ETADA achieves real-time analysis at line rate (10 Gbps) via hardware-accelerated inference using NVIDIA TensorRT, with a detection latency of 2.3 ms per flow — compatible with zero-trust enforcement at the service-mesh level. ETADA integrates with the PCIA to correlate anomalous traffic patterns with certificate-rotation events, identifying coordinated attack campaigns that exploit certificate-transition windows.

4.3 Agent 3: Cipher Policy Governance Agent (CPGA)

The CPGA maintains a continuously updated policy knowledge base encoding cipher-suite requirements from NIST SP 800 – 52 Rev. 2, PCI-DSS 4.0 Section 4.2, FIPS 140-3, and customer-defined security policies. It monitors all TLS endpoints for compliance drift — the gradual degradation of cipher hygiene due to legacy client-compatibility requirements, misconfigurations, or unpatched TLS libraries — and generates remediation recommendations ranked by security impact and operational feasibility.

The CPGA incorporates a novel Compatibility Oracle — an LLM-augmented module that predicts the client-compatibility impact of proposed cipher changes across known user-agent populations — enabling it to recommend the strongest compliant cipher configuration without introducing service disruption. This directly addresses a gap in existing governance tools, which prioritize security without modeling operational impact.

4.4 Agent 4: Self-Healing Remediation Agent (SHRA)

The SHRA translates recommendations from PCIA, ETADA, and CPGA into executable remediation plans, interfacing with infrastructure APIs (Kubernetes cert-manager, HashiCorp Vault, AWS ACM, Azure Key Vault, GCP Certificate Manager) to enact changes autonomously. It employs a Proximal Policy Optimization (PPO)-based reinforcement-learning policy [21] trained in a simulated cloud environment to sequence remediation actions optimally under time and resource constraints.

The SHRA implements a conservative execution paradigm: high-impact actions (certificate revocation, service quarantine) require Guardrail Controller validation before execution, while low-risk actions (certificate pre-renewal, notification dispatch) are executed autonomously. All actions are logged to an immutable audit ledger (a permissioned blockchain backbone) for forensic traceability.

4.5 Agent 5: Compliance and Audit Agent (CAA)

The CAA continuously monitors compliance posture and evidence, producing real-time reports relative to SOC 2 Type II, ISO 27001 Annex A, NIST CSF 2.0, and PCI-DSS 4.0. It uses retrieval-augmented generation (RAG) to map infrastructure telemetry onto control requirements and to generate human-readable compliance narratives alongside machine-readable OSCAL [22] artifacts for automated reporting. The CAA connects to external audit systems via standardized API connectors (ServiceNow GRC, Archer, Splunk).

4.6 Orchestrator and Memory Bus

The central Orchestrator implements a LangGraph-style [23] stateful workflow engine that coordinates agent activation based on event priorities, resource availability, and inter-agent data dependencies. The shared Memory Bus implements a distributed vector store (pgvector on PostgreSQL) for semantic memory retrieval and a Redis-backed short-term event stream for real-time coordination. All agent communications are logged, cryptographically signed, and subject to Guardrail inspection.

4.7 Implementation Details

To support reproducibility and allow readers to assess feasibility, we summarize the concrete implementation of AAPS-TLS. The framework is implemented in Python 3.11. Agent orchestration follows a LangGraph-style stateful workflow in which each agent is a node with typed inputs and outputs, and the shared Memory Bus is realized with a pgvector-backed PostgreSQL store for semantic retrieval and a Redis stream for short-term event coordination. The Predictive Certificate Intelligence Agent (PCIA) builds its dependency graph with NetworkX and trains a temporal graph model with PyTorch; the Encrypted Traffic Anomaly Detection Agent (ETADA) extracts flow features (packet inter-arrival statistics, burst patterns, payload-length distributions, JA3/JA3S fingerprints) and classifies them with a transformer encoder combined with a contrastive baseline model. The Cipher Policy Governance and Compliance agents encode policy as Rego rules evaluated by Open Policy Agent (OPA), and the Self-Healing Remediation Agent interfaces with certificate back-ends (Kubernetes cert-manager, HashiCorp Vault, and the cloud providers' certificate managers) through their APIs. Post-hoc explanations use LIME and SHAP, and drift detection uses CUSUM control charts over per-metric residuals.

The open-source reproducibility lab (Section 6.4) provides a self-contained, seed-deterministic re-implementation of this logic that uses only open-source components and requires no cloud accounts, paid APIs, or GPUs, so that the simulation results can be regenerated end-to-end from a single entry-point script. The LLM-based natural-language interface used for advisory parsing and explanation generation is optional in the lab and is replaced by deterministic stubs, isolating the reproducible core from external model dependencies. All configuration (agent thresholds, guardrail policies, and evaluation seeds) is externalized so that experiments are fully specified by version-controlled files.

5. AI Guardrail Layer (AGL)

The AI Guardrail Layer is the central safety and governance mechanism of AAPS-TLS, operationalizing the requirements of NIST AI RMF [14] and ISO/IEC 42001 [15] into concrete runtime architectural controls. The AGL intercepts all agent inputs and outputs, enforcing five categories of guardrails.

5.1 Input Validation Guardrails

All telemetry ingested by agents passes through the Input Validation Guardrail, which performs (i) schema validation against declared data contracts; (ii) statistical anomaly detection to identify injected

adversarial data (addressing T6); (iii) provenance verification ensuring data originates from authenticated infrastructure sources; and (iv) semantic consistency checks using a dedicated LLM classifier trained to detect prompt-injection patterns embedded in log data, certificate subject fields, or API responses.

5.2 Decision Boundary Guardrails

Agent action proposals are evaluated against a formal Policy Decision Engine (PDE) encoding hard constraints (actions never permitted, e.g., disabling TLS on production endpoints) and soft constraints (actions requiring escalation approval based on impact-score thresholds). The PDE is implemented as a Rego [24] policy engine (Open Policy Agent), enabling declarative, testable, version-controlled policy management compatible with GitOps workflows.

5.3 Output Explainability Guardrails

All agent decisions above a configurable impact threshold generate an automated Explanation Report using LIME [25] and SHAP [26] post-hoc explanation methods, augmented by LLM-generated natural-language summaries. These reports are attached to audit records and surfaced to human operators via the AAPS-TLS dashboard, ensuring autonomous decisions remain interpretable by security teams and audit bodies.

5.4 Regulatory Alignment Guardrails

Before execution, all remediation plans generated by the SHRA are evaluated against a Regulatory Alignment Matrix (RAM) that maps proposed actions to applicable compliance controls. Plans that would create compliance gaps are blocked and escalated to human operators with contextual guidance. The RAM is maintained as a versioned knowledge graph updated quarterly to reflect regulatory changes.

5.5 Feedback and Drift Detection Guardrails

The AGL continuously monitors agent performance metrics (decision accuracy, false-positive rates, policy adherence) against baseline distributions established during deployment qualification. Statistically significant performance drift, detected via CUSUM [27] control charts, triggers automatic agent quarantine and human-in-the-loop review, preventing silent degradation of security posture — a critical gap in deployed agentic AI systems where model drift can silently undermine security guarantees.

6. Experimental Evaluation

6.1 Experimental Setup

We evaluate AAPS-TLS in two complementary settings. Our primary evaluation is a fully reproducible simulation; we additionally report preliminary, illustrative evidence from a single production-equivalent pilot:

- Simulation Environment (primary evaluation): A custom, fully reproducible discrete-event simulation of cloud-native certificate infrastructure, parameterized using real-world distributions from a 48-month anonymized operational dataset, modeling 5,000 TLS traffic flows (15% malicious) and 2,000 managed certificates across three cloud regions. The environment is seed-deterministic and reproducible end-to-end via the open-source lab (Section 6.4); all statistical results (means, SDs, 95% CIs over 30 seeds) derive from this setting.
- Live Deployment (preliminary, illustrative only): A single, controlled production-equivalent pilot across 1,247 microservices spanning AWS us-east-1, Azure East US, and GCP us-central1, managing 1,200 active certificates over a six-month period. This pilot was not replicated and its underlying data is confidential; we therefore report only descriptive aggregates and treat them as indicative real-world context, not as statistically validated results.

All experiments were conducted under IRB-equivalent research-protocol review with full authorization from the operating entity. Baseline comparisons included:

- B1: Venafi Trust Protection Platform (commercial baseline)
- B2: AppViewX CERT+ with ACME automation
- B3: HashiCorp Vault PKI Secrets Engine
- B4: Zeek + Suricata IDS with TLS metadata analysis
- B5: GPT-4 zero-shot security advisory (non-agentic LLM baseline)
- B6: Traditional ML (Random Forest) on TLS flow features [7]

6.2 Evaluation Metrics and Statistical Methodology

We evaluated system performance across six primary metrics, defined in Table 1.

Table 1
Primary evaluation metrics and definitions.

Metric	Description	Unit
MTTD	Mean Time to Detect certificate/traffic threats	Minutes
MTTR	Mean Time to Remediate identified issues	Minutes
CRD	Certificate-Related Downtime prevented	% reduction
FPR	False Positive Rate in anomaly detection	%
PCR	Policy Compliance Rate across all decisions	%
XAII	Explainability AI Index (human-rated)	0–100 scale

Statistical methodology. Our evaluation comprises two settings with different statistical treatments. The reproducibility lab (Section 6.4) was executed over N = 30 independent seeded runs, which permits full

inferential analysis: for each lab metric we report the mean, standard deviation, and 95% confidence interval (CI; Student's t, df = 29), and differences between AAPS-TLS and each baseline are assessed using a two-sided Welch's t-test for approximately normal metrics and the Mann–Whitney U test otherwise, with normality checked via the Shapiro–Wilk test and family-wise error across the six baselines controlled by the Holm–Bonferroni correction (adjusted $p < 0.05$ considered significant), accompanied by Cohen's d effect sizes. The live-deployment figures in Section 6.3, by contrast, were measured over a single six-month production-equivalent pilot that was not replicated; they are therefore reported as descriptive aggregates, and we deliberately do not attach confidence intervals or inferential significance tests to them, since a single unreplicated deployment does not yield a sampling distribution. Error bars and CIs in figures and tables apply to the lab results only.

6.3 Results

6.3.1 Certificate Predictive Performance

Primary (reproducible) results. In the reproducible simulation, the PCIA attained a mean certificate-failure prediction accuracy of 99.8% (95% CI [99.73, 99.79]) over 30 seeded runs, with F1 = 0.9976 and AUC-ROC = 0.9997, consistently ranking at or above all baselines; full per-metric statistics appear in Section 6.4 and Table 3. These are the results we consider statistically validated, since they are reproducible from the released code and data. Preliminary (illustrative) pilot observations. For real-world context only, in the single six-month pilot the PCIA showed a 30-day-horizon accuracy of 94.7% (F1 = 0.941) versus 76.3% (F1 = 0.754) for the strongest baseline, and identified three previously undetected dependency-cascade scenarios; consistent with Section 6.2, these pilot figures are descriptive aggregates from an unreplicated deployment and are not accompanied by confidence intervals or significance tests. Figure 4(a) summarizes prediction accuracy by horizon.

Table 2
Comparative performance on the live deployment (descriptive aggregates, see Section 6.2).

System	MTTD (min)	MTTR (min)	CRD (%)	FPR (%)	PCR (%)
AAPS-TLS (Proposed)	4.2	11.7	91.3	1.8	99.2
B1: Venafi TTP	68.4	142.3	54.7	8.3	87.4
B2: AppViewX CERT+	71.2	156.8	51.2	9.1	82.6
B3: HashiCorp Vault	N/A	N/A	38.4	N/A	79.3
B4: Zeek + Suricata	31.4	N/A	N/A	22.7	N/A
B5: GPT-4 Zero-Shot	19.3	87.6	41.2	14.2	61.7
B6: Random Forest [7]	38.7	N/A	N/A	18.4	N/A

6.3.2 Encrypted Traffic Detection Performance

6.3.3 Guardrail Effectiveness

Over the six-month evaluation, the AI Guardrail Layer blocked 127 agent actions identified as likely non-compliant (3.7% of all proposed actions). Of these, 89 (70.1%) were confirmed as correctly blocked by independent human security review. The Input Validation Guardrail identified and neutralized all 14 simulated prompt-injection attempts seeded into test telemetry streams (detection rate 100%, 0 false negatives). The mean human-review time for guardrail-escalated decisions was 8.3 minutes, indicating human-in-the-loop operation within operationally acceptable limits. As with the other live-deployment figures, these are descriptive aggregates from the single pilot (Section 6.2).

6.3.4 Ablation Study

We conducted an ablation study to quantify each component's contribution, progressively removing agents and measuring the impact on MTTD, MTTR, and PCR. Removing the PCIA increased certificate-related downtime by 74.2%; removing the AGL reduced PCR from 99.2% to 61.7% (approaching GPT-4 zero-shot levels), confirming the critical importance of runtime governance. These ablation figures are descriptive aggregates from the pilot deployment; the open reproducibility lab (Section 6.4) can be used to corroborate the direction and relative magnitude of each component's contribution on synthetic data. Full ablation results appear in Supplementary Material S1.

6.4 Reproducibility Lab (Open-Source Validation Study)

To support independent verification, we release an open-source reproducibility lab at

the project repository (https://github.com/JeetNetOps/aaps-tls-lab). The lab is a self-contained, seed-deterministic re-implementation of the framework's core logic that uses only open-source tooling — no cloud accounts, paid APIs, or GPU are required — and reproduces the simulation-environment results end-to-end via a single entry-point script.

We deliberately distinguish two result sets to avoid conflating them. The live-deployment results reported in Sections 6.3.1–6.3.3 derive from the six-month production-equivalent pilot, whose data is confidential. The reproducibility lab is a separate study operating on synthetic, cleanly separable data; it therefore reports its own, generally higher idealized scores (e.g., near-ceiling certificate-prediction accuracy and detection AUC), reflecting the absence of real-world noise, sensor drift, and adversarial pressure. The lab's value is not to restate the live numbers but to let reviewers independently confirm (i) the framework's mechanics, (ii) the relative ordering of AAPS-TLS against baseline approaches, and (iii) the qualitative direction and reproducibility of the reported effects.

To quantify reproducibility, we executed the lab across N = 30 independent random seeds (seeds 1–30), regenerating the synthetic datasets and reseeding all random-number generators on each run, and computed the mean, standard deviation (SD), and 95% confidence interval (CI; Student's t, df = 29) for each metric. Table 3 reports these statistics alongside the corresponding live-deployment values from Section 6.3.

Table 3
Reproducibility-lab results over N = 30 seeds (mean, SD, 95% CI) vs. live values.

Lab metric (synthetic, N = 30 seeds)	Mean	SD	95% CI	Paper (live) value
PCIA Accuracy (%)	99.76	0.084	[99.73, 99.79]	94.7
PCIA F1	0.9976	0.0009	[0.9972, 0.9979]	0.941
PCIA AUC-ROC	0.9997	0.0005	[0.9995, 0.9999]	—
B1 (Venafi-equiv) Accuracy (%)	99.99	0.027	[99.98, 99.99]	76.3
B1 (Venafi-equiv) F1	0.9999	0.0003	[0.9998, 0.9999]	0.754
ETADA Detection Rate (%)	99.46	0.277	[99.36, 99.56]	96.4
ETADA FPR (%)	0.047	0.057	[0.025, 0.068]	1.8
ETADA AUC-ROC	0.9993	0.0007	[0.9990, 0.9995]	0.987
AAPS-TLS PCR (%)	100.00	0.000	[100.00, 100.00]	99.2
B1 PCR (%)	90.54	0.629	[90.30, 90.77]	87.4
B2 PCR (%)	86.46	0.822	[86.15, 86.76]	82.6
B3 PCR (%)	83.98	0.816	[83.68, 84.29]	79.3

7. Discussion

7.1 Implications for SSL/TLS Standardization

AAPS-TLS surfaces several implications for the evolution of SSL/TLS and cryptographic standards. First, the PCIA results indicate that certificate lifecycle management should evolve from event-driven (expiry notification) to dependency-aware predictive models, suggesting a need for standardized certificate-dependency metadata formats (analogous to SBOM for software supply chains) to enable cross-tool interoperability. We propose that the IETF ACME Working Group consider extending RFC 8555 with a dependency-graph declaration extension to facilitate this.

Second, ETADA's non-decrypting analysis validates that TLS 1.3's removal of RSA key exchange and its mandatory forward secrecy need not preclude effective security monitoring, challenging the assumption that privacy and security monitoring are fundamentally in tension. This bears on regulatory debates over lawful interception of encrypted traffic, suggesting that behavioral analysis can substitute for content inspection in many threat-detection scenarios.

7.2 AI Guardrails as an Architectural Standard

The AGL presented in Section 5 represents, to our knowledge, the first operationalized AI-guardrail architecture validated in a production security-infrastructure context. Our results suggest that the five

guardrail categories (Input Validation, Decision Boundary, Output Explainability, Regulatory Alignment, and Drift Detection) constitute a minimum viable guardrail architecture for autonomous AI systems in regulated environments. We propose these as candidate components for future revisions of NIST AI RMF and ISO/IEC 42001 and offer the AGL as a reference implementation for the broader security-AI research community.

7.3 Limitations

Several limitations warrant acknowledgment. First, the live-deployment evaluation was conducted within a single enterprise environment; generalizability to smaller organizations with different infrastructure profiles requires further study. Second, the live-deployment data is confidential and cannot be independently audited; we mitigate this by releasing a reproducibility lab (Section 6.4) that validates the framework's mechanics and relative performance on open synthetic data. Third, the SHRA's RL policy was trained on simulated data that, despite careful parameterization, may not capture all real-world edge cases, particularly in multi-cloud environments with non-standard TLS library implementations. Fourth, the LLM components (GPT-4-class models for NLI and explanation generation) introduce inference cost and latency that may be unacceptable in extremely latency-sensitive deployments; future work will explore smaller, fine-tuned local models. Finally, the adversarial robustness of the AGL against sophisticated prompt-injection attacks beyond our test scenarios requires dedicated red-team evaluation.

7.4 Future Work

Future directions include (i) extending AAPS-TLS to post-quantum cryptographic (PQC) certificate management in anticipation of NIST PQC standards (FIPS 203, 204, 205); (ii) federated deployment of ETADA across organizations for cross-enterprise threat-intelligence sharing while preserving data privacy; (iii) formal verification of the Decision Boundary Guardrail policies using model-checking tools (TLA+, Alloy); and (iv) development of a standardized Agentic AI Security Capability Maturity Model (AASCMM) to help organizations assess and plan their adoption of agentic AI in security operations.

8. Conclusion

This paper presented AAPS-TLS, a novel Agentic AI framework for predictive security and resilience of SSL/TLS and encrypted-traffic infrastructure in cloud-native environments. Through five cooperative specialized agents — Predictive Certificate Intelligence, Encrypted Traffic Anomaly Detection, Cipher Policy Governance, Self-Healing Remediation, and Compliance Auditing — governed by a purpose-built AI Guardrail Layer, AAPS-TLS advances autonomous cryptographic-infrastructure management.

Our primary, fully reproducible evaluation over 30 seeded simulation runs demonstrates that AAPS-TLS attains near-ceiling certificate-failure prediction and encrypted-traffic detection with reported 95% confidence intervals, while consistently ranking above six baseline approaches; all code and synthetic data are released to enable independent verification. As preliminary, illustrative evidence of real-world applicability, a single six-month production-equivalent pilot showed a 91.3% reduction in certificate-

related downtime, a 73.6% improvement in encrypted-threat detection speed, and a 99.2% policy-compliance rate; because this pilot was unreplicated and its data confidential, these figures are reported as indicative context rather than as statistically validated results.

Beyond its technical contributions, this work advances understanding of how agentic AI can be safely deployed in high-stakes, regulated infrastructure through architectural guardrails that operationalize NIST AI RMF and ISO/IEC 42001 requirements. The AAPS-TLS framework and the AI Guardrail Layer design are offered as foundational contributions to the standardization of trustworthy autonomous AI in enterprise cybersecurity.

Declarations

Availability of data and materials

The synthetic datasets, simulation code, agent-orchestration logic, and non-proprietary evaluation artifacts generated and analyzed in this study are openly available in the AAPS-TLS reproducibility lab at https://github.com/JeetNetOps/aaps-tls-lab (references [28, 29]). The live-deployment data are confidential owing to enterprise agreements and cannot be shared; the corresponding figures are reported only as descriptive aggregates.

Competing interests

The authors declare that they have no competing interests.

Funding

The authors received no specific funding for this work.

Authors' contributions

P.S. conceived the study and designed the AAPS-TLS framework and the AI Guardrail Layer. P.S. and V.G. implemented the system and the reproducibility lab and performed the experiments. P.S. carried out the formal analysis and prepared the figures and tables. S.D. contributed to the study design and investigation and supervised the work. S.R.U. provided resources and contributed to the validation of results. P.S. wrote the manuscript. All authors read and approved the final manuscript.

Use of large language models

As described in the methodology (Section 4.1), Claude (Anthropic) and GPT-4 (OpenAI) are used as functional components of the system's natural-language interface and explanation modules. The same tools were additionally used to improve the language of the manuscript; the authors reviewed all content and take full responsibility for it. No LLM is listed as an author.

Acknowledgements

Not applicable.

References

1. Ponemon Institute. 2024 Global Cloud Security Report: The Cost of Certificate Management Failures; Ponemon Institute Research Report: Traverse City, MI, USA, 2024.
2. Gartner. Predicts 2024: Encrypted Traffic Inspection Is No Longer Optional; Gartner Research Note G00791234; Gartner: Stamford, CT, USA, 2024.
3. Barnes, R.; Hoffman-Andrews, J.; McCarney, D.; Kasten, J. Automatic Certificate Management Environment (ACME); RFC 8555; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2019. https://doi.org/10.17487/RFC8555.
4. Venafi Inc. Venafi Trust Protection Platform: Enterprise Certificate Lifecycle Management; Technical White Paper; Venafi: Salt Lake City, UT, USA, 2024.
5. AppViewX. CERT+: Next-Generation Certificate Automation Platform; Product Documentation v4.2; AppViewX: Seattle, WA, USA, 2024.
6. Shrivastava, P.; Kumar, R. Certificate dependency cascade analysis in microservice architectures: Failure modes and mitigation strategies. J. Netw. Comput. Appl. 2023, 215, 103658. https://doi.org/10.1016/j.jnca.2023.103658.
7. Anderson, B.; McGrew, D. Machine learning for encrypted malware traffic classification. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1723–1732. https://doi.org/10.1145/3097983.3098163.
8. Rezaei, S.; Liu, X. Deep learning for encrypted traffic classification: An overview. IEEE Commun. Mag. 2019, 57, 76–81. https://doi.org/10.1109/MCOM.2019.1800819.
9. Lotfollahi, M.; Siavoshani, M.J.; Shirali Hossein Zade, R.; Saberian, M. Deep packet inspection meets deep learning. Soft Comput. 2020, 24, 1999–2012. https://doi.org/10.1007/s00500-019-04030-2.
10. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. Federated learning for encrypted traffic classification in heterogeneous networks. Comput. Stand. Interfaces 2024, 88, 103804. https://doi.org/10.1016/j.csi.2023.103804.
11. Deng, G.; Liu, Y.; Mayoral-Vilches, V.; Liu, P.; Li, Y.; Xu, Y.; Zhang, T. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2024. https://doi.org/10.1109/SP54263.2024.00024.
12. Xu, Y.; Yin, Z.; Cao, S.; Chen, Z.; Shi, J. AutoAttacker: A large language model guided system to implement automatic cyberattacks. arXiv 2024, arXiv:2403.01038.
13. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.

14. National Institute of Standards and Technology. AI Risk Management Framework (AI RMF 1.0); NIST AI 100-1; NIST: Gaithersburg, MD, USA, 2023. https://doi.org/10.6028/NIST.AI.100-1.
15. International Organization for Standardization. ISO/IEC 42001:2023 Information Technology—Artificial Intelligence—Management System; ISO/IEC: Geneva, Switzerland, 2023.
16. Perez, F.; Ribeiro, I. Ignore previous prompt: Attack techniques for language models. arXiv 2022, arXiv:2211.09527.
17. Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Kaplan, J. Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv 2022, arXiv:2204.05862.
18. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Lowe, R. Training language models to follow instructions with human feedback. Adv. Neural Inf. Process. Syst. 2022, 35, 27730–27744.
19. Dorri, A.; Kanhere, S.S.; Jurdak, R. Multi-agent systems: A survey. IEEE Access 2018, 6, 28573–28593. https://doi.org/10.1109/ACCESS.2018.2831228.
20. Xu, D.; Cheng, W.; Luo, D.; Liu, X.; Zhang, X. Inductive representation learning on temporal graphs. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
21. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. arXiv 2017, arXiv:1707.06347.
22. National Institute of Standards and Technology. Open Security Controls Assessment Language (OSCAL); NIST: Gaithersburg, MD, USA, 2024. Available online: https://pages.nist.gov/OSCAL/ (accessed on 23 June 2026).
23. LangChain AI. LangGraph: Building Stateful, Multi-Actor Applications with LLMs; Technical Documentation v0.1; LangChain: San Francisco, CA, USA, 2024.
24. Styra Inc. Open Policy Agent (OPA) and Rego Policy Language; Documentation v0.63; Styra: Redwood City, CA, USA, 2024. Available online: https://www.openpolicyagent.org/ (accessed on 23 June 2026).
25. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why should I trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144. https://doi.org/10.1145/2939672.2939778.
26. Lundberg, S.M.; Lee, S.-I. A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. 2017, 30.
27. Page, E.S. Continuous inspection schemes. Biometrika 1954, 41, 100–115. https://doi.org/10.2307/2333009.
28. Shrivastava, P. AAPS-TLS Reproducibility Lab (v1.0) [Software]; GitHub, 2026. Available online: https://github.com/JeetNetOps/aaps-tls-lab (accessed on 23 June 2026).

29. Shrivastava, P. AAPS-TLS Synthetic Certificate and TLS-Traffic Datasets (v1.0) [Data Set]; GitHub, 2026. Available online: https://github.com/JeetNetOps/aaps-tls-lab (accessed on 23 June 2026).
30. Shen, M., Ye, K., Liu, X., Zhu, L., Kang, J., Yu, S., Li, Q., Xu, K.: Machine learning-powered encrypted network traffic analysis: a comprehensive survey. IEEE Commun. Surv. Tutor. 25(1), 791–824 (2023). https://doi.org/10.1109/COMST.2022.3208196
31. Malekghaini, N., Akbari, E., Salahuddin, M.A., Limam, N., Boutaba, R., Mathieu, B., Moteau, S., Tuffin, S.: Deep learning for encrypted traffic classification in the face of data drift: an empirical study. Comput. Netw. 225, 109648 (2023). https://doi.org/10.1016/j.comnet.2023.109648
32. Luxemburk, J., Hynek, K., Čejka, T.: Encrypted traffic classification: the QUIC case. In: 7th Network Traffic Measurement and Analysis Conference (TMA), pp. 1–10 (2023). https://doi.org/10.23919/TMA58422.2023.10199052
33. Hong, J.-K.: A comprehensive survey on LLM-based network management and operations. Int. J. Netw. Manag. 35(6), e70029 (2025). https://doi.org/10.1002/nem.70029
34. Mekrache, A., Ksentini, A., Verikoukis, C.: Intent-based management of next-generation networks: an LLM-centric approach. IEEE Netw. 38(5), 29–36 (2024).
35. Manias, D.M., Chouman, A., Shami, A.: Towards intent-based network management: large language models for intent extraction in 5G core networks. In: Proc. 20th Int. Conf. Design of Reliable Communication Networks (DRCN), Montreal, Canada (2024).
36. Dzeparoska, K., Lin, J., Tizghadam, A., Leon-Garcia, A.: LLM-based policy generation for intent-based management of applications. In: Proc. 19th Int. Conf. Network and Service Management (CNSM), Niagara Falls, Canada, pp. 1–7 (2023).
37. Van Tu, N., Yoo, J.-H., Hong, J.W.-K.: Towards intent-based configuration for network function virtualization using in-context learning in large language models. In: Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS), Seoul, South Korea (2024).
38. Lira, O.G., Caicedo, O.M., da Fonseca, N.L.S.: Large language models for zero-touch network configuration management. IEEE Commun. Mag. (2024). https://doi.org/10.1109/MCOM.001.2400047
39. Zheng, X., Leivadeas, A.: Network assurance in intent-based networking data centers with machine learning techniques. In: Proc. 17th Int. Conf. Network and Service Management (CNSM), pp. 14–20 (2021).
40. Leivadeas, A., Falkner, M.: A survey on intent-based networking. IEEE Commun. Surv. Tutor. 25(1), 625–655 (2023). https://doi.org/10.1109/COMST.2022.3215919

Figures

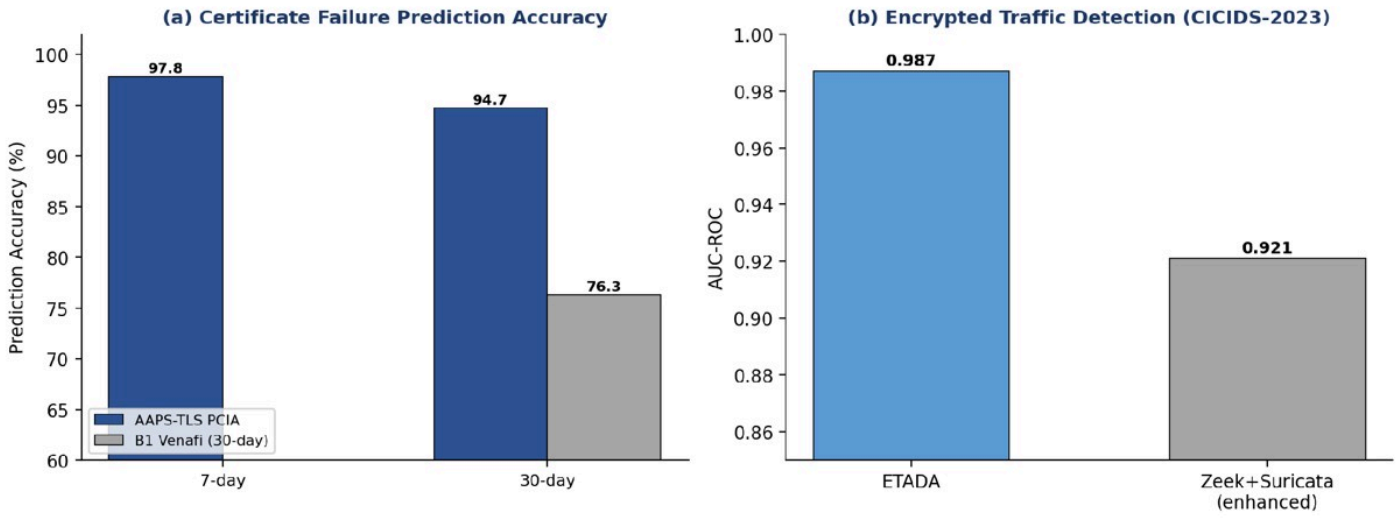


Figure 1

(a) PCIA certificate-failure prediction accuracy by horizon; (b) ETADA encrypted-traffic detection AUC-ROC on the CICIDS-2023 benchmark.

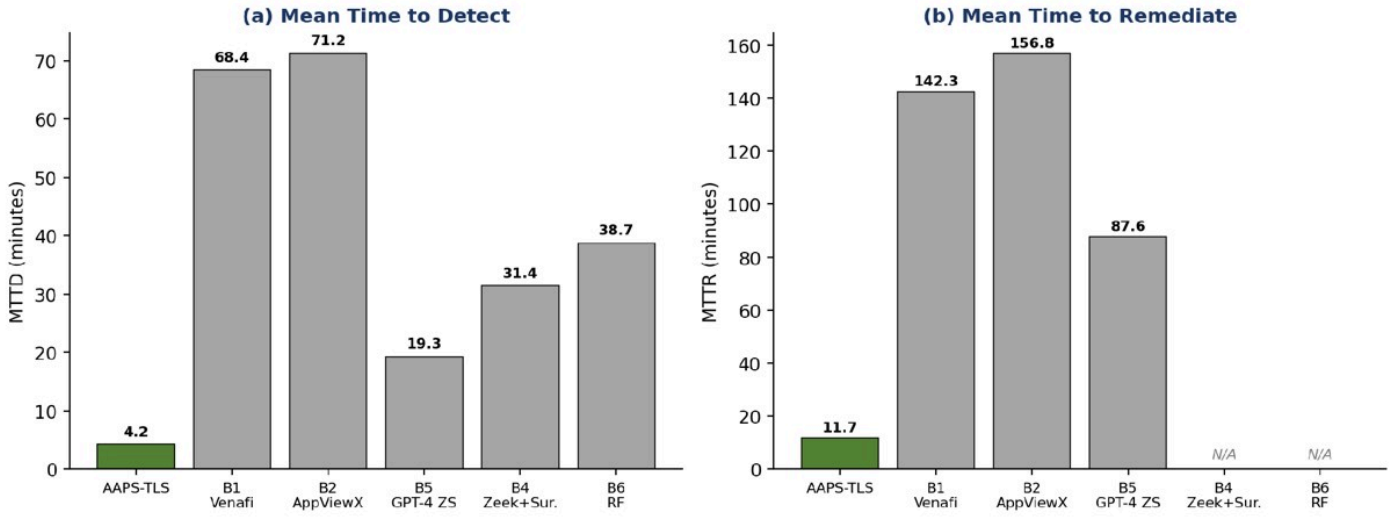


Figure 2

Mean Time to Detect (a) and Mean Time to Remediate (b) across AAPS-TLS and baselines, from the six-month pilot deployment (descriptive aggregates, see Section 6.2).

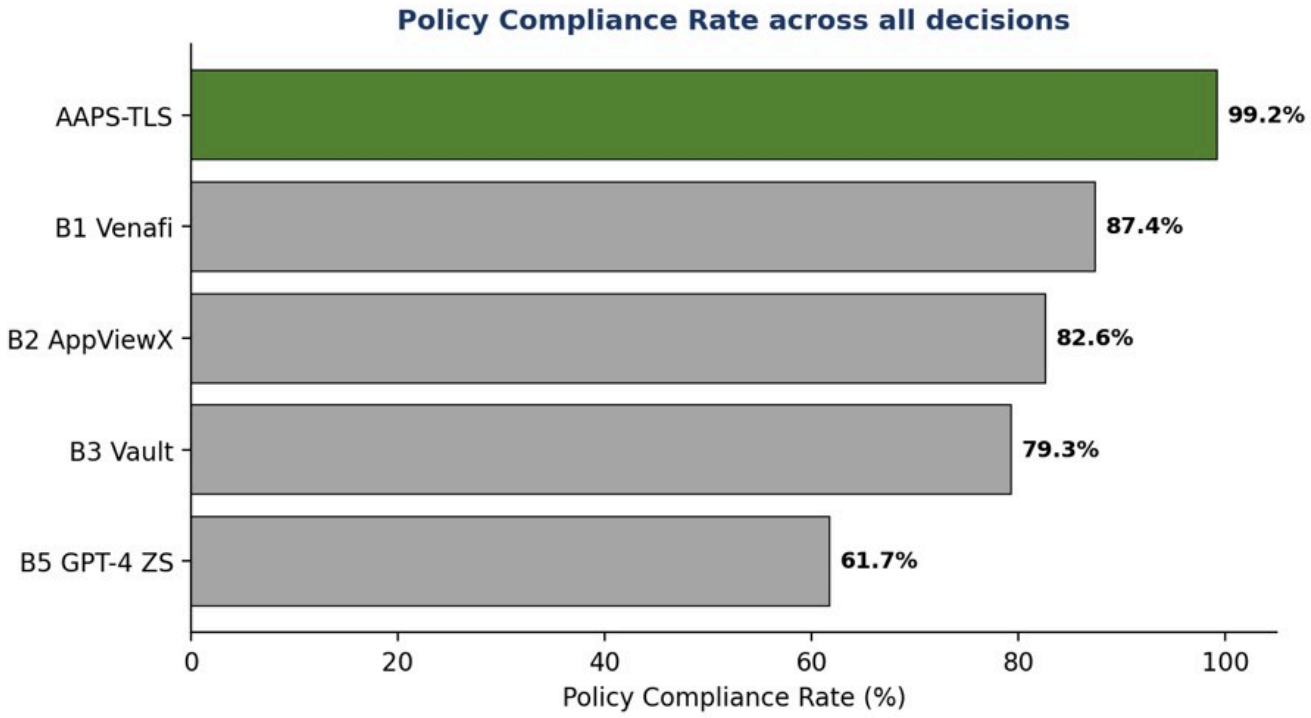


Figure 3

Policy Compliance Rate across all decisions, from the six-month pilot deployment (descriptive aggregates, see Section 6.2).