

# Compliance as a Trust Metric

Wenbo Wu<sup>[0009–0002–3937–0124]</sup> and George Konstantinidis<sup>[0000–0002–3962–9303]</sup>

University of Southampton, Southampton, UK  
`{wenbo.wu, g.konstantinidis}@soton.ac.uk`

**Abstract.** Trust and Reputation Management Systems (TRMSs) are critical for the modern web, yet their reliance on subjective user ratings or narrow Quality of Service (QoS) metrics lacks objective grounding. Concurrently, while regulatory frameworks like GDPR and HIPAA provide objective behavioral standards, automated compliance auditing has been limited to coarse, binary (pass/fail) outcomes. This paper bridges this research gap by operationalizing regulatory compliance as a quantitative and dynamic trust metric through our novel automated compliance engine (ACE). ACE first formalizes legal and organizational policies into a verifiable, obligation-centric logic. It then continuously audits system event logs against this logic to detect violations. The core of our contribution is a quantitative model that assesses the severity of each violation along multiple dimensions, including its *Volume*, *Duration*, *Breadth*, and *Criticality*, to compute a fine-grained, evolving compliance score. We evaluate ACE on a synthetic hospital dataset, demonstrating its ability to accurately detect a range of complex HIPAA and GDPR violations and produce a nuanced score that is significantly more expressive than traditional binary approaches. This work enables the development of more transparent, accountable, and resilient TRMSs on the Web.

**Keywords:** Compliance · Quantification · Trust · Reputation.

## 1 Introduction

Trust is the foundation that enables interactions among unfamiliar users in distributed networks [30,36] like data markets [12,26,23]. To facilitate this, TRMSs act as intermediaries that collect and process trust signals to calculate a reputation for each network node, enabling more reliable interaction experiences [18,36]. However, traditional TRMSs typically rely on subjective user ratings or context-specific QoS metrics [32,37,14]. In parallel, the digital landscape is increasingly governed by formal regulations like GDPR [10] and HIPAA [8], which provide an objective and authoritative foundation for evaluating an entity’s compliance level [31]. In regulated industries, demonstrable compliance can be the foundational proxy for trustworthiness. Despite this, the systems we use to measure digital trust have been slow to adapt, leaving compliance auditing as a research gap in modern trust evaluation [36].

Organizations generate data usage logs to record system operations, and authorities typically audit these logs manually to detect violations. This reliance

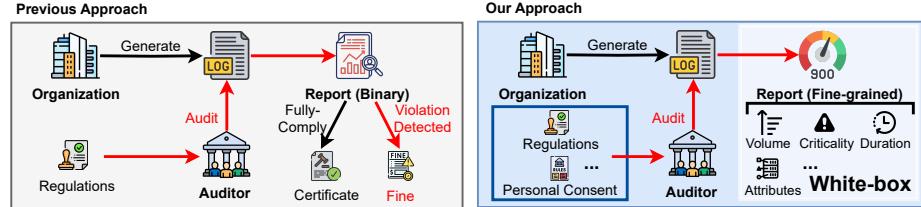


Fig. 1: Binary Auditing vs. Our Fine-grained Auditing

on manual auditing is problematic, as the process is time-consuming, costly, and error-prone [1]. While automated compliance checking has emerged as a field of study, its methods are incapable of dynamic trust assessment. Existing work (*e.g.*, [1,3,35]) predominantly produces binary outcomes—an entity is either compliant or not—which fails to capture the nuanced spectrum of adherence (see the left half of Fig. 1). Furthermore, these audit results are typically used for static, single-point-in-time reporting rather than as a live, evolving metric for long-term reputation tracking. This leaves their potential to enrich trust models largely underexplored.

To bridge this gap, we argue that regulatory compliance should be treated not as a binary check, but as a quantitative and dynamic trust metric. Accordingly, this paper introduces a novel "white-box" (*i.e.*, the use of explicit, human-readable logic rules rather than opaque neural network embeddings) Automated Compliance Engine (ACE) that provides fine-grained, automated compliance assessment (see the right half of Fig. 1). This ensures that every generated compliance score is backed by a verifiable, deterministic audit trail, allowing auditors to trace exactly why a specific action was flagged as a violation. ACE first translates unstructured legal regulations into verifiable logic, which enables a continuous audit of system event logs to detect violations. It then quantifies the degree of adherence along multiple dimensions, producing a nuanced compliance score. Finally, this score is designed for seamless integration as an authoritative new dimension within a TRMS, enriching the overall trustworthiness evaluation. ACE has broad applicability in domains, including but not limited to data markets [12,26,23] and decentralized finance [38,2], offering a more robust and transferable standard for establishing trust in digital environments. Our primary contributions are:

- **A Novel Framework for Compliance-Based Trust:** We introduce ACE, a white-box framework that operationalizes regulatory compliance as a verifiable and dynamic trust metric for TRMSs (Sec. 2).
- **Formalization of Policies into Verifiable Logic:** We present a methodology to translate complex legal and organizational policies into a machine-verifiable, obligation-centric logic, enabling automated and continuous auditing (Sec. 3.1, 3.2, 3.3, and Appendix A and B).
- **A Multi-Dimensional Quantitative Scoring Model:** We design a quantitative model that transforms discrete violation data into a fine-grained,

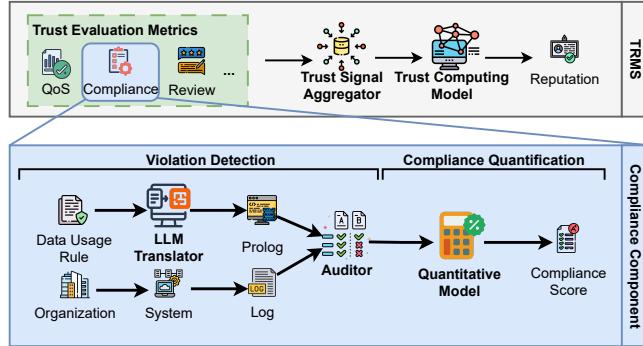


Fig. 2: Fine-Grained Compliance Component for TRMS

continuous compliance score by assessing multiple dimensions of violations, including its volume, duration, breadth, and criticality (Sec. 3.4, 3.5).

- **An Empirical Evaluation of Feasibility and Effectiveness:** We implement and evaluate an open-source prototype of our framework on a synthetic hospital dataset, demonstrating its accuracy in detecting violations and the superior expressiveness of its scoring model compared to traditional binary approaches (Sec. 4).

## 2 System Overview

The overall TRMS design incorporates compliance as a key, quantifiable trust metric, as illustrated in Fig. 2. The framework is designed to generate a holistic reputation score from diverse trust signals, which are classified as explicit (*e.g.*, textual Reviews and ratings) and implicit (*e.g.*, QoS and, crucially, Compliance) [36]. These varied signals are continuously collected and processed by a Trust Signal Aggregator, which applies strategies like time-decay weighting to prepare the data. The prepared data is then fed into a Trust Computing Model, which uses statistical or machine learning algorithms to calculate a final, unified reputation score [33]. This score serves as a key reference, enabling users to make informed, evidence-based decisions when engaging with other entities.

The primary innovation of this framework lies in its fine-grained compliance component, which transforms compliance from a simple ‘checkbox’ into a verifiable, data-driven trust metric. Its operation is divided into two main phases: Violation Detection and Compliance Quantification. The process begins by preparing two key inputs. First, all relevant data handling actions performed by an Organization are captured in an immutable system Log, which serves as objective evidence of its behavior. Second, the governing Data Usage Rules, derived from sources like GDPR [13], patient consent agreements [19], and HIPAA-based policies [27,7,29], are formalized. Our framework leverages an LLM Translator to convert these high-level policies into machine-executable Prolog rules.

In the Violation Detection phase, an automated Auditor systematically compares the event Log against the formalized Prolog rules to identify any disrepan-

cies or violations. The audit’s output—a stream of identified violations—is then passed to the Compliance Quantification phase. Here, a Quantitative Model assesses these violations based on predefined criteria, such as their frequency and the volume of data affected, to calculate a final numerical **Compliance Score**.

By processing real-world operational data against formal rules, this component provides a robust and objective measure of an entity’s adherence to its commitments. The fine-grained compliance score is then integrated into the main TRMS as a key trust metric, directly linking an entity’s reputation to its actual compliance posture. Ultimately, this makes the overall TRMS more objective, transparent, and resilient to manipulation compared to systems that rely solely on subjective feedback.

### 3 The Compliance Component

At the core of our system is the compliance component (*i.e.*, ACE) formally detecting and quantifying policy violations. This section details the theoretical ground of ACE: (1) the *Compliance Policy Logic* that defines the language of our rules; (2) the *Compliance Verification Semantics* that define the meaning of compliance; (3) the *Compliance Model and Violation Semantics*; (4) the *Quantifying Compliance Violations*; and (5) the *Computing Compliance Score* model that produces a fine-grained compliance score.

#### 3.1 Compliance Policy Logic

The foundation of our compliance framework is a policy language grounded in a decidable fragment of first-order logic [1,13]. This choice provides an expressive, unambiguous syntax for codifying complex, real-world rules. Unlike traditional authorization logics that focus on specifying permissions (*i.e.*, what a user *may* do) [1], our logic is tailored to express **obligations** and **constraints** [5] (*i.e.*, what *must* be true in a given situation). The goal is to formalize policies into verifiable logic to automate compliance auditing and build quantifiable trust. Following [1,13], the formal syntax of the policy language is defined as:

**Definition 1 (Policy Language Syntax).** *The grammar of our compliance policy language is formally defined by the components:*

$$\begin{aligned}
 & \text{(Types)} \quad \sigma ::= \text{principal} \mid \text{resource} \mid \text{role} \mid \dots \\
 & \text{(Constants)} \quad c ::= \text{Alice} \mid \text{Bob\_PHI} \mid \text{Doctor} \mid \dots \\
 & \text{(Variables)} \quad x, y, z ::= \text{A set of typed variables } \mathcal{V} \\
 & \text{(Terms)} \quad t ::= c \mid x \quad \text{where } c \text{ and } x \text{ are of the same type } \sigma \\
 & \text{(Predicates)} \quad P ::= \text{has\_role} \mid \text{is\_doctor\_of} \mid \text{read} \mid \dots \\
 & \text{(Atoms)} \quad \alpha ::= P(t_1, \dots, t_n) \\
 & \text{(Formulas)} \quad \phi, \psi ::= \alpha \mid \phi_1 \wedge \phi_2 \quad (\text{Conjunctive Formulas}) \\
 & \text{(Rule)} \quad \rho ::= (\forall \mathbf{x}.(\phi \supset \psi), C) \\
 & \text{(Policy)} \quad \Pi ::= \{\rho_1, \dots, \rho_m\}
 \end{aligned}$$

*Types, Terms, and Predicates.* **Types** ( $\sigma$ ) partition entities into distinct domains like **principals** (users, services) and **resources** (data, objects). **Terms** ( $t$ ) are the fundamental arguments in logical statements, representing either specific **constants** ( $c$ ) like the user *Alice*, or **variables** ( $x, y$ ) that stand for any entity of a given type. **Predicates** ( $P$ ) declare named relations over these terms, such as  $\text{is\_doctor\_of}(p, q)$ .

*Atoms and Formulas.* An **atom** ( $\alpha$ ) is the application of a predicate to a sequence of terms, forming the most basic provable statement (e.g.,  $\text{has\_role}(Alice, Doctor)$ ). In our framework, **formulas** ( $\phi, \psi$ ) are constructed as conjunctions ( $\wedge$ ) of these atoms. This structure is expressive enough for a wide range of practical compliance rules while maintaining desirable computational properties similar to Datalog [16].

*Compliance Rules: Triggers and Constraints.* The core of our language is the **compliance rule** ( $\rho$ ). A policy  $\Pi$  is a finite set of such rules. Each rule is a tuple containing a logical formula and a numeric **Rule Criticality** ( $C \in [0, 1]$ ). The formula itself,  $\forall \mathbf{x}.(\phi \supset \psi)$ , establishes an obligation: i) The **universal quantifier**  $\forall$  (read “for all”) binds the variables in the vector  $\mathbf{x}$ , making the rule a general statement that applies universally. For brevity, we often assume variables are universally quantified over the scope of the entire rule. ii) The **Trigger Condition** ( $\phi$ ) is a formula that acts as the rule’s premise. It describes a specific event or state, such as an action being performed. When facts corresponding to the trigger are observed, the rule’s obligation is invoked. iii) The **implication symbol** ‘ $\supset$ ’ separates the trigger from the constraint. iv) The **Required Constraint** ( $\psi$ ) is a formula describing a condition that *must* hold true whenever the trigger is satisfied.

For instance, a hospital policy rule can be:  $(\forall p_1, p_2, r.(\text{read}(p_1, r) \wedge \text{is\_phi}(r) \supset \text{has\_role}(p_1, doctor) \wedge \text{is\_doctor\_of}(p_1, p_2) \wedge \text{owns\_phi\_record}(p_2, r)), 0.9)$ . This rule does not grant permission to read; instead, it asserts that *if* a read on PHI occurs (trigger), then the entity performing the read must have the ‘doctor’ role and is the doctor of the resource owner (constraint). A failure to meet this constraint is a compliance violation.

### 3.2 Compliance Verification Semantics

The verification semantics give formal meaning to our policy language by defining how the truth of a formula is derived from a set of known facts. This process, known as entailment, provides the mechanism for checking the trigger and constraint conditions of a compliance rule. We define this through an **entailment judgment**:  $\mathcal{K}, l \vdash \phi$ . This judgment is read as: “The formula  $\phi$  is entailed (or is provably true) from the knowledge base  $\mathcal{K}$  in the context of a log entry  $l$ .” The log entry  $l = (\alpha_{req}, \tau)$  is crucial as it provides the timestamp  $\tau$  of the event being verified, allowing for temporally-aware fact checking.

**Definition 2 (Semantic Inference Rules).** *The entailment relation  $\vdash$  is defined as the smallest relation closed under the following inference rules. These*

rules specify how to prove positive, conjunctive formulas, which is the foundation for verifying our compliance rules.

**Axiom for Facts.** This rule is the bridge between the stored data and the logic. It states that a ground atom is provably true if a corresponding fact exists in the knowledge base  $\mathcal{K}$  and its timestamped validity interval  $[T_{start}, T_{end}]$  contains the timestamp  $\tau$  of the event being checked.

$$\frac{(P(c_1, \dots, c_n), T_{start}, T_{end}) \in \mathcal{K} \quad l = (\dots, \tau) \quad T_{start} \leq \tau \leq T_{end}}{\mathcal{K}, l \vdash P(c_1, \dots, c_n)} \text{ (AXIOM)}$$

**Conjunction Introduction.** This rule defines the semantics of the logical AND ( $\wedge$ ). It states that a conjunction of formulas is true if, and only if, a proof can be derived for each individual formula. This rule is used to verify the multi-part conditions often found in rule triggers and constraints.

$$\frac{\mathcal{K}, l \vdash \phi_1 \quad \dots \quad \mathcal{K}, l \vdash \phi_n}{\mathcal{K}, l \vdash \phi_1 \wedge \dots \wedge \phi_n} \text{ (AND-INTRO)}$$

**Universal Instantiation (** $\forall$ **-Elimination).** This rule is the bridge between a general policy and a specific, observable event. It states that if a universally quantified formula is provably true, then any specific instance of that formula, created by substituting the variable with a constant  $c$  from the system's domain, is also provably true.

$$\frac{\mathcal{K}, l \vdash \forall x. \phi(x)}{\mathcal{K}, l \vdash \phi(c)} \text{ ( $\forall$ -ELIM)}$$

**Implication Elimination (** $\supset$ **-Elimination).** Commonly known as Modus Ponens, this rule allows the system to derive new facts. If a rule's premise  $\phi$  is proven true, and the rule itself ( $\phi \supset \psi$ ) is established, then the conclusion  $\psi$  can be inferred as a new fact. While not strictly required for our violation semantics (Def. 3.4), it enables rule chaining and a more powerful reasoning engine.

$$\frac{\mathcal{K}, l \vdash \phi \supset \psi \quad \mathcal{K}, l \vdash \phi}{\mathcal{K}, l \vdash \psi} \text{ ( $\supset$ -ELIM)}$$

These semantic rules provide the formal, operational engine for automated compliance verification. To check a specific action recorded in a log entry  $l$  against a rule, the system uses this proof system. As detailed in the next subsection, a violation is detected if, for some substitution  $\theta$ , the system can successfully build a proof for the trigger condition  $(\mathcal{K}, l \vdash \phi\theta)$  but fails to build a proof for the required constraint  $(\mathcal{K}, l \not\vdash \psi\theta)$ .

### 3.3 Compliance Model and Violation Semantics

To quantify compliance, we first need a formal model to define what constitutes a violation. We extend our logical framework from defining permissions to specifying **obligations** and **constraints**. A violation, therefore, is not merely an un-permitted action, but an action that breaks a required constraint.

**Definition 3 (Compliance Model).** A system is a tuple  $(\mathcal{K}, \mathcal{L}, \Pi)$ , where:

- $\mathcal{K}$  is the **Knowledge Base**, is the system’s central repository of ground-truth information, acting as the authoritative source of facts against which compliance is measured. It is not a static database but is formally defined as a set of timestamped facts.
- $\mathcal{L}$  is the **Access Log** is the system’s complete, time-ordered, and immutable record of all actions, serving as the objective evidence consumed by the automated Auditor. The conceptual log can comprise a **staff activity log** for internal operations and a **patient request log** for external compliance-related actions like data access or erasure requests.
- $\Pi$  is the **Compliance Policy**, a set of compliance rules. Each rule  $\rho \in \Pi$  is a pair  $(\forall \mathbf{x}.(\phi \supset \psi), C)$ , where  $C$  is rule criticality.

Unlike the authorization logic which defines what a principal *may* do, this logic defines what *must* be true when a certain action occurs. A violation occurs when the trigger condition is met but the required constraint is not.

**Definition 4 (Violation Semantics).** An access log entry  $l = (\alpha_{req}, \tau)$  constitutes a **violation** of a rule  $\rho = (\forall \mathbf{x}.(\phi \supset \psi), C)$  if there exists a ground substitution  $\theta$  that aligns the log entry with the rule’s trigger, but the rule’s constraint is not met. Formally, a violation occurs if:

$$\underbrace{\mathcal{K}, l \vdash \phi\theta}_{\text{Trigger is met}} \quad \wedge \quad \underbrace{\mathcal{K}, l \not\vdash \psi\theta}_{\text{Constraint is not met}} \quad (1)$$

where the judgment  $\vdash$  is derived using the enforcement semantics (Sec. 3.2). Typically, the trigger formula  $\phi$  includes an action atom that unifies with  $\alpha_{req}$  from the log entry.

**Theorem 1 (Violation as Logical Inconsistency).** The detection of a violation for a log entry  $l$  under a policy  $\Pi$  is equivalent to finding a logical inconsistency between the compliance rule  $\rho \in \Pi$  and the observed state of the world (i.e., the log entry  $l$  combined with the facts in  $\mathcal{K}$ ).

*Proof Sketch:* The proof follows from the definition of a violation (Def. 4). The detection algorithm finds a substitution  $\theta$  such that the premise of an implication  $(\phi\theta)$  is true, while the conclusion  $(\psi\theta)$  is false. This configuration,  $\phi\theta \wedge \neg\psi\theta$ , is a direct contradiction of the rule’s assertion that  $\phi\theta \supset \psi\theta$  must hold for all substitutions. The algorithm is a direct implementation of this semantic check, ensuring no false positives.  $\square$

### 3.4 Quantifying Compliance Violations

A binary violation flag lacks the granularity needed for a dynamic trust metric. To capture the true magnitude of non-compliance, we introduce a quantitative model that operates over discrete, configurable time windows,  $W$  (e.g., weekly or

monthly). This periodic approach assesses a principal's behavior within a given time window, providing a basis for a continuously evolving compliance score.

**Periodic Violation Metrics.** For each principal  $p$  and each time window  $W$ , all detected violations are first grouped by the specific rule  $\rho_i \in \Pi$  that was broken. From these groupings, we derive a set of raw metrics that characterize the scope and persistence of the non-compliant behavior for that rule.

**Definition 5 (Per-Rule Violation Metrics).** *For each rule  $\rho_i$  violated by a principal  $p$  during a window  $W$ , we compute the following metrics:*

- **Volume ( $M_V$ ):** *The total number of unique resources (the number of specific rows/records, e.g., 50 patient files) affected by violations of rule  $\rho_i$ . This captures the scale of the non-compliance.*
- **Duration ( $M_T$ ):** *The total time elapsed from the first to the last logged violation of rule  $\rho_i$  within the window. This measures the persistence of the failure to comply.*
- **Breadth ( $M_B$ ):** *The number of distinct resource attributes (the number of columns/fields affected, e.g., accessing 'DoB' and 'Address' = breadth of 2) involved in violations of rule  $\rho_i$ . This assesses the scope of the incident across different data categories.*

**Normalizing Violation Metrics.** The raw metrics exist on different, unbounded scales. To combine them meaningfully, each metric is transformed into a normalized score on a  $[0, 1]$  scale using a negative exponential function. This function models the principle of diminishing impact, where each additional unit of violation contributes less to the overall severity than the one before it.

**Definition 6 (Normalized Severity Components).** *For each raw metric  $M \in \{M_V, M_T, M_B\}$ , its normalized score  $S$  is given by:*

$$S(M) = 1 - e^{-(M/\alpha)}$$

where  $\alpha$  is a configurable scaling parameter unique to each metric ( $\alpha_V, \alpha_T, \alpha_B$ ). This parameter defines the characteristic value at which the severity is considered significant. A small  $\alpha$  denotes high sensitivity to minor violations, while a large  $\alpha$  requires a greater violations to yield a high score.

**Calculating Violation Magnitude.** With normalized components, we can compute a unified magnitude score for each rule violation. We employ a weighted geometric mean to capture the synergistic effect of multiple dimensions of non-compliance.

**Definition 7 (Per-Rule Violation Magnitude).** *The Magnitude  $M_{i,p,W}$  for a rule  $\rho_i$  violated by principal  $p$  in window  $W$  is the weighted geometric mean of its normalized severity components, this is theoretically grounded in the "synergistic effect" of non-compliance:*

$$M_{i,p,W} = (S_V^{w_V} \cdot S_T^{w_T} \cdot S_B^{w_B})$$

where the weights  $w_j \geq 0$  and  $\sum w_j = 1$ . We employ a weighted geometric mean rather than a linear sum to mathematically enforce that "widespread, persistent, and broad violations are disproportionately more severe than the linear sum of their parts". This ensures that a low score in any single dimension significantly dampens the overall magnitude, preventing the system from over-penalizing trivial incidents while accurately capturing the scale of complex violations.

### 3.5 Computing the Compliance Score

The final stage of our model aggregates the per-rule violation data into a single, evolving compliance score for each principal. This is achieved by first calculating a total severity penalty for the current period, and then updating a long-term, time-decaying cumulative penalty.

**Per-Period Severity.** First, the magnitude of each rule violation is scaled by the rule's intrinsic importance. These scaled values are then summed to find the total penalty incurred during the period.

**Definition 8 (Total Per-Period Severity).** *The total severity score  $Sev_{p,W}$  for a principal  $p$  in a window  $W$  is the sum of the severity scores of all rules they violated. The severity of a single rule violation is its magnitude multiplied by its predefined criticality  $C_i \in [0, 1]$ .*

$$Sev_{p,W} = \sum_{i \in \text{violated rules}} (C_i \cdot M_{i,p,W})$$

*This additive approach is justified as violations of distinct rules represent independent and cumulative failures of compliance.*

**Evolving Principal Compliance Score.** Unlike static, single-window scores, our final compliance score evolves over time, reflecting a principal's entire history while giving more weight to recent actions. This provides a path to redemption for principals who improve their behavior.

**Definition 9 (Compliance Score).** *The compliance score for a principal  $p$  at the end of period  $W_k$  is derived from a time-decaying cumulative penalty. The penalty is updated recursively:*

$$\text{Penalty}_k = (\text{Penalty}_{k-1} \cdot e^{-\lambda}) + Sev_{p,W_k}$$

*where  $\text{Penalty}_{k-1}$  is the cumulative penalty from the previous period and  $\lambda$  is a "decay constant" that determines how quickly past violations are forgiven, which serves as the mathematical realization of a "Path to Redemption". The final compliance score is then calculated by mapping this unbounded penalty to the range  $[0, 1]$  using the hyperbolic tangent function:*

$$\text{Comp}(p, W_k) = 1 - \tanh(\text{Penalty}_k)$$

*A score of 1 indicates perfect compliance, while a score approaching 0 indicates a history of severe and/or recent non-compliance.*

**Theorem 2 (Boundedness of Score).** *The final Compliance Score  $\text{Comp}(p, W_k)$  is guaranteed to be in the range  $[0, 1]$ .*

*Proof Sketch:* The hyperbolic tangent function,  $\tanh(x)$ , is bounded in the range  $[-1, 1]$  for any real input  $x$ . Since the Penalty score is always non-negative,  $\tanh(\text{Penalty}_k)$  is bounded in  $[0, 1]$ . Consequently,  $1 - \tanh(\text{Penalty}_k)$  is also guaranteed to be in the range  $[0, 1]$ , making it a well-formed input for TRMSs.

□

## 4 Experiments

To validate our proposed framework, we conduct a series of experiments designed to evaluate the accuracy and performance of our auditor, ACE, in detecting a variety of compliance violations within a simulated hospital environment. Also, we demonstrate that our fine-grained quantitative model produces a compliance score that is more expressive and informative in distinguishing the severity of different violations when compared to traditional baseline models.

### 4.1 Experimental Setup

The ACE auditor uses SWI-Prolog 9.0 as its logic engine, integrated via the `pyswip` library. The auditing and quantitative scoring were performed using Python 3.9. All experiments were conducted on an Apple machine with an M1 Max chip and 32GB of memory.

**Policy Corpus:** We formalize five key compliance rules relevant to a hospital scenario, as detailed in Section 3 and Appendix B. These include three rules based on GDPR data subject rights (Art. 15, 17, 18), one authorization rule and one minimum necessary rule based on the principles of HIPAA.

**Dataset:** We develop a synthetic data generator designed as an augmented derivation of the MIMIC-III dataset [17] structure. This generator produces a hospital dataset comprising a **Knowledge Base**, a **Staff Activity Log**, and a **Patient Request Log** that mirrors the specific attributes, schema, and relational complexity (*e.g.*, patient-doctor assignments, billing codes, and PHI timestamps) found in the MIMIC-III critical care database. While the specific entries are generated to ensure privacy, this alignment ensures that the data's structural complexity is representative of real-world healthcare environments. The dataset includes three types of principals: doctor, patient, and billing clerk. We adopted this augmented approach rather than utilizing raw clinical logs because existing real-world datasets lack the explicit ground-truth violation labels required for precise validation. Our generator is capable of injecting specific, verifiable violation scenarios (*e.g.*, complex GDPR "Right to be Forgotten" failures) with corresponding labels, establishing a rigorous ground truth for our analysis.

### 4.2 Evaluation of Violation Detection

We evaluate ACE's violation-detection accuracy and performance using the synthetic datasets. Each dataset is constructed with 5% labeled violating entries

Table 1: Performance of the ACE Auditor.

Log Entries	Runtime Throughput Peak Memory			Runtime Throughput Peak Memory		
	<i>Staff Activity Log</i>			<i>Patient Request Log</i>		
5,000	50.173 s	99.655 entries/s	81.66 MB	57.076 s	87.603 entries/s	78.33 MB
10,000	112.006 s	89.281 entries/s	81.5 MB	112.699 s	88.732 entries/s	80.5 MB
50,000	596.302 s	83.85 entries/s	117.11 MB	659.441 s	75.822 entries/s	125.0 MB
100,000	1299.936 s	76.927 entries/s	179.92 MB	1291.577 s	77.425 entries/s	159.62 MB
200,000	2654.017 s	75.357 entries/s	221.16 MB	2666.071 s	75.017 entries/s	231.64 MB
500,000	6363.378 s	78.575 entries/s	413.72 MB	6688.031 s	74.76 entries/s	485.39 MB

under the single-rule-per-violation constraint. The auditor is executed on both staff-activity and patient-request logs across dataset sizes from 5k to 500k.

**Detection accuracy.** Across all 10 runs (five staff logs and five patient logs), the auditor reported exactly the same number of rule instances as the number of labeled violations. This corresponds to perfect recall on the labeled violation set (100% of labeled violations were detected) and confirms that the ACE auditor can accurately detect all the violations defined in the policy rules.

**Scalability and resource use.** Table 1 summarizes runtime, throughput (entries per second), and peak resident memory observed for each test. Runtime grows approximately linearly with the number of rows. The patient-request series exhibits similar behavior. Throughput declines modestly with scale indicating a roughly constant per-row processing cost with modest overhead growth. Peak memory usage increases with dataset size but remains comfortably below 500 MB for our largest runs.

**Implications.** The empirical findings show that ACE achieves perfect detection while exhibiting near-linear runtime scaling in the number of log entries. The measured throughput indicate that ACE can handle medium-to-large offline audit workloads on commodity hardware. The primary bottleneck is overall per-row processing time (driven by KB assertions, Prolog query overhead, and I/O). For larger deployments or near-real-time auditing, straightforward optimizations are available: batching assertions, reducing KB re-loading, parallelizing independent record checks, or moving hot predicates to a compiled Prolog module.

### 4.3 Evaluation of the Quantitative Scoring

Beyond the core accuracy of violation detection, we also evaluate the effectiveness and expressiveness of our quantitative scoring model. The following case studies are designed to demonstrate how the fine-grained score provides a more nuanced and dynamic measure of compliance compared to traditional approaches.

**Differentiating Violation Severity.** This experiment evaluates the model’s primary claim of expressiveness. The goal is to demonstrate that the ACE produces a nuanced score of non-compliant scenarios that fundamentally differ in traditional models are expected to fail.

We compare the ACE compliance score against two standard baselines: a binary model and a simple count-based model. The evaluation is conducted

Table 2: Comparison of model scores across four violation scenarios. A lower compliance score indicates a more severe assessment of the principal’s behavior.

Scenario	#violations	ACE	Binary	Count
A: Low Impact	1	<b>0.553</b>	0	0.99
B: High Volume	50	<b>0.364</b>	0	0.50
C: High Criticality	2	<b>0.478</b>	0	0.98
D: High Duration	25	<b>0.420</b>	0	0.75

across four crafted scenarios for a single principal, `doctor_A`, within a 30-day window. The baselines are defined as follows:

- **Binary Model:**  $S_{\text{binary}} = 0$  if any violation is detected, otherwise 1. This model captures only the presence or absence of non-compliance.
- **Count-Based Model:**  $S_{\text{count}} = 1 - (N_{\text{violations}}/N_{\text{total}})$ , where  $N_{\text{total}}$  is the total number of log entries audited. This model’s assessment is driven solely by the frequency of violation events.

Four scenarios were designed to isolate the impact of different violation types:

**Scenario A (Low Impact):** A single, isolated violation of a medium-criticality rule (`hipaa_auth_control`,  $C = 0.8$ ) affecting one resource for one day. **Scenario B (High Volume):** A large-scale violation of the same rule, affecting 50 unique resources over a two-day period. **Scenario C (High Criticality):** A single, severe violation of a high-criticality rule (`gdpr_art17_erasure`,  $C = 0.95$ ), where a statutory 30-day deadline was missed. **Scenario D (High Duration):** A persistent violation involving repeated unauthorized access to a single resource, occurring once per day for 25 consecutive days.

The comparative results, presented in Table 2, highlight the superior expressiveness of the ACE model. As anticipated, the Binary model is unable to differentiate between any of the non-compliant scenarios, assigning a score of 0 to all. The Count-Based model, while offering some differentiation, produces a misleading assessment; for instance, it rates the highly critical GDPR violation (Scenario C) as less severe than the persistent access in Scenario D, and nearly identical to the low-impact Scenario A, because it is insensitive to rule criticality.

In contrast, the ACE model provides a more nuanced evaluation. It correctly identifies the single low-impact event (A) as the least severe. It also produces a distinct ranking for the other scenarios based on the current parameterization, which is ordered by severity as: B (High Volume)  $\succ$  D (High Duration)  $\succ$  C (High Criticality)  $\succ$  A (Low Impact). This demonstrates that the ACE score is sensitive to multiple dimensions of non-compliance. Furthermore, the model’s parameterization allows for ranking to be tuned to reflect specific organizational priorities. For example, by increasing the weight of the criticality component, an organization can configure the model to ensure that Scenario C is rated as the most severe, thereby aligning the quantitative score with its specific risk posture. **Dynamic Score Evolution.** This case study demonstrates the dynamic, time-aware nature of the ACE score by tracking the evolving compliance of multiple principals over an extended period. The goal is to show how the score provides

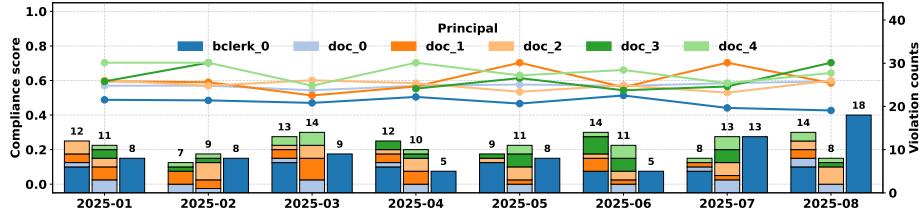


Fig. 3: Monthly compliance trends and per-month rule violations

a nuanced, continuous measure of trustworthiness that is more expressive than simple violation counts. We simulated an eight-month activity log for six principals (one billing clerk and five doctors) and calculated their compliance scores at the end of each month. The goal is to visualize how the compliance score degrades in response to violations and, crucially, recovers when compliant behavior is restored, demonstrating the model's "path to redemption" feature.

Figure 3 plots the monthly compliance scores (left axis, line graph) and the raw violation counts (right axis, bar chart) for each principal. The results highlight several key features of our model:

- **Dynamic Responsiveness:** The compliance scores for all principals fluctuate monthly, directly responding to their actions within each period. This confirms the model’s ability to serve as a live, dynamic metric for tracking behavior over time.
- **Nuance Beyond Raw Counts:** The score is not a simple inverse of the violation count. For instance, in month 2025-08, `bclerk_0` and `doc_1` have similar violation counts (14 and 13, respectively), yet their compliance scores differ significantly (approximately 0.4 vs. 0.6). This demonstrates that the ACE score incorporates the severity of violations rather than just frequency.
- **Long-Term Trend Tracking:** The visualization effectively captures long-term behavioral trends. For example, `doc_3` shows a general upward trend in their compliance score after an initial dip, while `bclerk_0`’s score remains consistently lower, reflecting a persistent pattern of more severe non-compliance. This makes the score suitable for long-term trust and reputation tracking, where historical behavior and possibilities for recovery are essential.

**Sensitivity Analysis of Model Parameters.** This experiment analyzes the impact of the model’s key tunable parameters on the final compliance score (the other parameters are analyzed in Appendix C), demonstrating its flexibility. The results in Figure 4 show how the model can be configured to align with different organizational priorities and risk tolerances by adjusting the decay constant ( $\lambda$ ) and the normalization scaling factors ( $\alpha$ ).

*Analysis of the Decay Constant ( $\lambda$ )* We analyzed the model’s "path to redemption" feature by varying the decay constant  $\lambda$ , which controls how quickly past violations are forgiven. The left panel of Fig. 4 plots the compliance score’s recovery over 100 days since the last violation for three different  $\lambda$  values. A higher value of  $\lambda$  ( $\lambda = 1.0$ ) corresponds to high forgiveness, with the score recovering

Fig. 4: Parameter sensitivity:  $\lambda$  (left) and  $\alpha$  (right).

to near-perfection in approximately 10 days. Conversely, a lower value ( $\lambda = 0.1$ ) represents low forgiveness, with the score recovering much more slowly. This confirms that the  $\lambda$  parameter effectively controls the system’s memory and allows it to be tuned to be more or less forgiving of past violations.

*Analysis of the Scaling Factor ( $\alpha$ )* We then analyzed the sensitivity of the severity calculation to the scaling factor  $\alpha$ . The right panel of Fig. 4 shows the normalized severity score for fixed raw violations ( $M_V = 20, M_T = 10, M_B = 3$ ) as their corresponding scaling factors,  $\alpha_V, \alpha_T, \alpha_B$ , are adjusted. A small  $\alpha$  ( $\alpha = 5$ ) results in a high severity score, making the system highly sensitive to even a moderate number of violations. A large  $\alpha$  ( $\alpha = 50$ ) yields a much lower score, indicating that a greater number of violations would be needed to be considered severe.

## 5 Related Work

Our research is positioned at the intersection of three key areas: TRMSs, automated compliance checking, and the quantitative compliance assessment.

**Trust and Reputation Management.** Trust management has a long history in distributed systems [21,22,36]. Early work focused on formal, rule-based systems for trust negotiation [3] and managing trust assertions on the Semantic Web [28]. Subsequent research integrated these policy-based approaches with reputation mechanisms that aggregate behavioral feedback over time [4]. Other related works explore how to measure trust in data-driven systems by discovering conformance constraints [11] or how user-facing privacy explanations can foster end-user trust [6]. While foundational, these systems typically rely on subjective user ratings, pre-negotiated service level agreements, or narrow QoS metrics [25,15]. They often lack a continuous, objective, and verifiable signal grounded in adherence to formal regulations. Our work fills this gap by introducing verifiable regulatory compliance as a new, high-integrity input for TRMSs.

**Automated Compliance Checking.** The field of automated compliance checking focuses on algorithmically verifying system behavior against a set of policies [31,34]. Systems like PrivGuard have made significant strides in making privacy regulation compliance easier to check [35]. Researchers have also developed formal methods for automating audits, even when system logs are incomplete [1]. Other approaches have leveraged Natural Language Processing (NLP) to create datasets for identifying regulation compliance in unstructured software privacy

policies [41,40,42]. However, the primary output of these systems is typically a binary decision: an entity is either compliant or not. These tools are designed for internal audits or enforcement actions rather than for generating a dynamic, public-facing trust signal. Our framework builds upon the outputs of such systems, transforming their discrete violation data into a continuous, fine-grained score suitable for reputation management.

**Quantitative Compliance Assessment.** The most closely related research shares our motivation for moving beyond binary compliance. Zhang *et al.* proposed a method for quantitative evaluation of compliance levels in the context of safety regulations [39]. Chen *et al.* [9] proposed the concept of quantifying compliance, their work is strictly limited to predictive monitoring using "black-box" machine learning models to forecast temporal delays (a single dimension) with respect to policy deadlines. While these works pioneer the concept of quantitative compliance, our contribution is distinct and complementary. As we solve fundamentally different problems using distinct technical architectures. We introduce a comprehensive, end-to-end framework that begins with a formal, obligation-centric logic for policy definition. We then propose a novel, multi-dimensional quantification model and a time-decaying aggregation mechanism specifically designed to produce an evolving score. Crucially, we are the first to explicitly design this quantitative compliance score as a modular and integrable trust metric for enhancing modern TRMSs.

## 6 Discussion

In this section, we discuss 1) the practical applicability of the ACE within distributed ecosystems, 2) analyze deployment considerations regarding log integration, and 3) demonstrate the model's robustness against strategic exploitation.

**Use Cases.** While traditional regulatory frameworks often rely on static, infrequent audits (*e.g.*, annual GDPR certifications [20]), modern distributed environments require real-time operational trust visibility. Continuous scoring adds distinct value in dynamic ecosystems, such as decentralized data markets [12,24,26]. In these environments, data sellers must establish trust that data buyers will continuously comply with data usage agreements throughout the transaction lifecycle, rather than relying on a compliance snapshot from a previous audit cycle. ACE bridges this gap by providing a live, evolving metric, enabling participants to make decisions based on the current compliance posture of a counterpart.

**Log Conversion.** A practical concern for deploying ACE is the overhead associated with converting heterogeneous institutional logs into the system's verifiable format. We clarify that this process constitutes a standard Extract-Transform-Load (ETL) task. The integration requires a one-time schema mapping to align internal system logs with ACE's Prolog-based fact structure. Furthermore, the risk of conversion errors or data misalignment is significantly mitigated by the strict typing defined in our *Policy Language Syntax* (Def. 1). This syntax acts as a validation layer, rejecting malformed inputs or type mismatches before the auditing process begins, thereby ensuring the integrity of the evaluation data.

**Robustness Against Strategic Exploitation.** A requirement for any quantitative auditing system is adversarial resilience, where a malicious actor might attempt to tactically violate rules to mask severe issues. The mathematical structure of the ACE model can inherently prevent such exploitation through three key mechanisms: *i*) Unlike average-based systems where "good" behavior dilutes "bad" grades, ACE utilizes an Additive Penalty Model (Def. 8). Severity is the sum of weighted violations; thus, committing additional minor violations increases the penalty stack rather than averaging it down. *ii*) The Rule Criticality parameter ensures high-impact violations dominate the score regardless of lower-impact compliance. This prevents entities from masking critical breaches behind a volume of trivial compliant actions. *iii*) The system counters "splitting" attacks by grouping related events into single maximal instances based on timestamp (Defs. 5, 6). Furthermore, recovery relies solely on Time Decay (Def. 9); entities cannot recover reputation with compliant actions but must cease violations to rebuild trust over time.

## 7 Conclusion

In this paper, we introduced ACE, a novel white-box framework that transforms regulatory compliance from a static, binary assessment into a quantitative and dynamic trust metric. By formalizing legal and organizational policies into a verifiable, obligation-centric logic, our system continuously audits event logs to compute a fine-grained compliance score. This score, derived from multiple dimensions, assessing violation volume, duration, breadth, and criticality, offers an expressive measure of trustworthiness than traditional approaches.

While our evaluation demonstrated the ACE's accuracy and the nuanced output of our quantitative model, we acknowledge several limitations that pave the way for future research. First, the process of translating complex, often ambiguous legal text into formal logic currently requires significant manual effort from domain experts. Second, our validation was conducted on a synthetic dataset; testing the ACE's robustness against the noise and inconsistencies of real-world operational logs is a vital next step. Finally, the current performance of our prototype is well-suited for periodic audits, and further optimization would be necessary for its application in near-real-time trust-aware systems.

Future work will proceed along several key directions to address these limitations. We plan to evaluate the ACE on large-scale, anonymized enterprise datasets to validate its generalizability and real-world efficacy. To mitigate the policy formalization bottleneck, we will explore fully-automated techniques to the translation process. A key theoretical extension will be to enhance our model to handle correlated violations, where a single underlying fault may breach multiple rules, to ensure the resulting score accurately reflects the root cause without unfair penalization. By pursuing these extensions, we aim to build more transparent, accountable, and resilient trust ecosystems on the web.

**Acknowledgments.** This work was partially funded by the Horizon Europe project DATAPACT (101189771) and the UKRI Horizon Europe guarantee funding scheme for the Horizon Europe projects RAISE (101058479) and UPCAST (101093216).

## References

1. Bichhawat, A., Fredrikson, M., Yang, J.: Automating audit with policy inference. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF). pp. 1–16 (2021). <https://doi.org/10.1109/CSF51468.2021.00001>
2. Bodo, B., De Filippi, P.: Trust in context: the impact of regulation on blockchain and defi. Regulation & Governance (2024). <https://doi.org/https://doi.org/10.1111/rego.12637>
3. Bonatti, P., De Coi, J., Olmedilla, D., Sauro, L.: A rule-based trust negotiation system. IEEE Transactions on Knowledge and Data Engineering **22**(11), 1507–1520 (2010). <https://doi.org/10.1109/TKDE.2010.83>
4. Bonatti, P., Duma, C., Olmedilla, D., Shahmehri, N.: An integration of reputation-based and policy-based trust management. In: W9: The Semantic Web and Policy Workshop (SWPW). vol. 2, p. 136. Citeseer (2007)
5. Breaux, T.D., Vail, M.W., Anton, A.I.: Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In: 14th IEEE International Requirements Engineering Conference (RE’06). pp. 49–58. IEEE (2006). <https://doi.org/https://doi.org/10.1109/RE.2006.68>
6. Brunotte, W., Specht, A., Chazette, L., Schneider, K.: Privacy explanations – a means to end-user trust. Journal of Systems and Software **195**, 111545 (2023). <https://doi.org/https://doi.org/10.1016/j.jss.2022.111545>, <https://www.sciencedirect.com/science/article/pii/S0164121222002217>
7. Byun, J.W., Li, N.: Purpose based access control for privacy protection in relational database systems. The VLDB Journal **17**, 603–619 (2008). <https://doi.org/https://doi.org/10.1007/s00778-006-0023-0>
8. Centers for Medicare & Medicaid Services: The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at <http://www.cms.hhs.gov/hipaa/> (1996)
9. Chen, Q., Rinderle-Ma, S., Wen, L.: Beyond yes or no: Predictive compliance monitoring approaches for quantifying the magnitude of compliance violations (2025), <https://arxiv.org/abs/2502.01141>
10. European Parliament and Council of the European Union: General Data Protection Regulation (GDPR) (2018), <https://gdpr-info.eu/>
11. Fariha, A., Tiwari, A., Radhakrishna, A., Gulwani, S., Meliou, A.: Conformance constraint discovery: Measuring trust in data-driven systems. In: Proceedings of the 2021 International Conference on Management of Data. pp. 499–512 (2021). <https://doi.org/https://doi.org/10.1145/3448016.3452795>
12. Fernandez, R.C., Subramaniam, P., Franklin, M.J.: Data market platforms: Trading data assets to solve data problems. Proceedings of the VLDB Endowment **13**(11) (2020)
13. Garg, D., Jia, L., Datta, A.: Policy auditing over incomplete logs: theory, implementation and applications. In: Proceedings of the 18th ACM conference on Computer and communications security. pp. 151–162 (2011)
14. Ghose, A., Koliadis, G.: Auditing business process compliance. In: International Conference on Service-Oriented Computing. pp. 169–180. Springer (2007). [https://doi.org/https://doi.org/10.1007/978-3-540-74974-5\\_14](https://doi.org/https://doi.org/10.1007/978-3-540-74974-5_14)

15. Goo, J., Huang, C.D.: Facilitating relational governance through service level agreements in it outsourcing: An application of the commitment–trust theory. *Decision Support Systems* **46**(1), 216–232 (2008). <https://doi.org/https://doi.org/10.1016/j.dss.2008.06.005>
16. Huang, S.S., Green, T.J., Loo, B.T.: Datalog and emerging applications: an interactive tutorial. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. pp. 1213–1216 (2011). <https://doi.org/https://doi.org/10.1145/1989323.1989456>
17. Johnson, A.E., Pollard, T.J., Shen, L., Lehman, L.W.H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., Mark, R.G.: Mimic-iii, a freely accessible critical care database. *Scientific data* **3**(1), 1–9 (2016). <https://doi.org/https://doi.org/10.1038/sdata.2016.35>
18. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. *Decision support systems* **43**(2), 618–644 (2007). <https://doi.org/https://doi.org/10.1016/j.dss.2005.05.019>
19. Konstantinidis, G., Holt, J., Chapman, A.: Enabling personal consent in databases. *Proceedings of the VLDB Endowment* **15**(2), 375–387 (2021). <https://doi.org/https://doi.org/10.14778/3489496.3489516>
20. Lachaud, E.: What gdpr tells about certification. *Computer Law & Security Review* **38**, 105457 (2020)
21. Liu, H., Han, D., Li, D.: Behavior analysis and blockchain based trust management in vanets. *Journal of Parallel and Distributed Computing* **151**, 61–69 (2021). <https://doi.org/https://doi.org/10.1016/j.jpdc.2021.02.011>
22. Liu, Y., Wang, J., Yan, Z., Wan, Z., Jäntti, R.: A survey on blockchain-based trust management for internet of things. *IEEE internet of Things Journal* **10**(7), 5898–5922 (2023). <https://doi.org/https://doi.org/10.1109/JIOT.2023.3237893>
23. Ma, Y., Jiang, X., Wu, E.W., Ibáñez, L.D., Shi, J.: Model-based data markets: a multi-broker game theoretic approach. In: 2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 2293–2301. IEEE (2024)
24. Ma, Y., Jiang, X., Wu, W., Yang, Z., Ibáñez, L.D.: Mixture-of-experts based model market. In: VLDB 2025 Workshop: 3rd Data EConomy Workshop (DEC). VLDB Endowment (2025), proceedings of the VLDB 2025 Workshop
25. Manuel, P.: A trust model of cloud computing based on quality of service. *Annals of Operations Research* **233**(1), 281–292 (2015). <https://doi.org/https://doi.org/10.1007/s10479-013-1380-x>
26. Nguyen, T.L., Nguyen, L., Hoang, T., Bandara, D., Wang, Q., Lu, Q., Xu, X., Zhu, L., Chen, S.: Blockchain-empowered trustworthy data sharing: Fundamentals, applications, and challenges. *ACM Computing Surveys* **57**(8), 1–36 (2025). <https://doi.org/https://doi.org/10.1145/3718082>
27. Pereira, A.L., Muppavarapu, V., Chung, S.M.: Role-based access control for grid database services using the community authorization service. *IEEE Transactions on Dependable and Secure Computing* **3**(2), 156–166 (2006). <https://doi.org/https://doi.org/10.1109/TDSC.2006.26>
28. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) *The Semantic Web - ISWC 2003*. pp. 351–368. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). [https://doi.org/https://doi.org/10.1007/978-3-540-39718-2\\_23](https://doi.org/https://doi.org/10.1007/978-3-540-39718-2_23)
29. Servos, D., Osborn, S.L.: Current research and open problems in attribute-based access control. *ACM Computing Surveys (CSUR)* **49**(4), 1–45 (2017). <https://doi.org/https://doi.org/10.1145/3007204>

30. Su, Z., Liu, L., Li, M., Fan, X., Zhou, Y.: Reliable and resilient trust management in distributed service provision networks. *ACM Transactions on the Web (TWEB)* **9**(3), 1–37 (2015). [https://doi.org/https://doi.org/10.1145/2754934](https://doi.org/10.1145/2754934)
31. Truong, N.B., Sun, K., Lee, G.M., Guo, Y.: Gdpr-compliant personal data management: A blockchain-based solution. *IEEE Transactions on Information Forensics and Security* **15**, 1746–1761 (2019)
32. Wahab, O.A., Bentahar, J., Otrok, H., Mourad, A.: A survey on trust and reputation models for web services: Single, composite, and communities. *Decision Support Systems* **74**, 121–134 (2015). <https://doi.org/https://doi.org/10.1016/j.dss.2015.04.009>
33. Wang, J., Jing, X., Yan, Z., Fu, Y., Pedrycz, W., Yang, L.T.: A survey on trust evaluation based on machine learning. *ACM Computing Surveys (CSUR)* **53**(5), 1–36 (2020). <https://doi.org/https://doi.org/10.1145/3408292>
34. Wang, L., Guan, Z., Chen, Z., Hu, M.: Enabling integrity and compliance auditing in blockchain-based gdpr-compliant data management. *IEEE Internet of Things Journal* **10**(23), 20955–20968 (2023)
35. Wang, L., Khan, U., Near, J., Pang, Q., Subramanian, J., Soman, N., Gao, P., Low, A., Song, D.: PrivGuard: Privacy regulation compliance made easier. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 3753–3770. USENIX Association, Boston, MA (Aug 2022), <https://www.usenix.org/conference/usenixsecurity22/presentation/wang-lun>
36. Wu, W., Konstantinidis, G.: Trust and reputation in data sharing: A survey (2025), <https://arxiv.org/abs/2508.14028>
37. Yu, Y., Liu, S., Guo, L., Yeoh, P.L., Vucetic, B., Li, Y.: CrowdR-fbc: A distributed fog-blockchains for mobile crowdsourcing reputation management. *IEEE Internet of Things Journal* **7**(9), 8722–8735 (2020). <https://doi.org/https://doi.org/10.1109/JIOT.2020.2996229>
38. Zetsche, D.A., Arner, D.W., Buckley, R.P.: Decentralized finance. *Journal of Financial Regulation* **6**(2), 172–203 (2020)
39. Zhang, B., Chu, Z., Cheng, L., Zou, N.: A quantitative safety regulation compliance level evaluation method. *Safety Science* **112**, 81–89 (2019). <https://doi.org/https://doi.org/10.1016/j.ssci.2018.10.016>, <https://www.sciencedirect.com/science/article/pii/S0925753518301413>
40. Zhang, J., El-Gohary, N.M.: Semantic nlp-based information extraction from construction regulatory documents for automated compliance checking. *Journal of computing in civil engineering* **30**(2), 04015014 (2016). [https://doi.org/https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000346](https://doi.org/https://doi.org/10.1061/(ASCE)CP.1943-5487.0000346)
41. Zhao, K., Yu, L., Zhou, S., Li, J., Luo, X., Chiu, Y.F.A., Liu, Y.: A fine-grained Chinese software privacy policy dataset for sequence labeling and regulation compliant identification. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (eds.) *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. pp. 10266–10277. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022). <https://doi.org/10.18653/v1/2022.emnlp-main.700>, <https://aclanthology.org/2022.emnlp-main.700>
42. Zhou, Y.C., Zheng, Z., Lin, J.R., Lu, X.Z.: Integrating nlp and context-free grammar for complex rule interpretation towards automated compliance checking. *Computers in Industry* **142**, 103746 (2022). <https://doi.org/https://doi.org/10.1016/j.compind.2022.103746>

## A Illustrative Example of Semi-Automated Policy Formalization

This section provides a concrete, step-by-step walkthrough of our semi-automated, human-in-the-loop process for formalizing a compliance obligation. We use the HIPAA Authorization Control rule as a running example to demonstrate the workflow from a plain-text policy segment to a final, expert-verified logical rule. This process is designed to accelerate development while ensuring correctness and mitigating the risk of model-induced errors such as hallucination or logical misinterpretation.

**Example: HIPAA Authorization Control (hipaa\_auth)**

### Step 1: Isolate Policy Text

The process begins with a specific, actionable requirement from a compliance document.

*"A principal acting in the role of a 'doctor' is only authorized to read a Patient Health Information (PHI) record if they are the designated physician for the patient who owns that record."*

### Step 2: Generate Candidate Rule with LLM

This text is fed into a Large Language Model using a structured prompt that includes the schema definition and few-shot examples. The goal is to generate a candidate rule that captures the core logic. Below is a simplified representation of the prompt structure.

The LLM processes this prompt and generates the following candidate rule:

#### LLM-Generated Candidate Rule:

```
violation('hipaa_auth', Doctor, PHI_Record) :-  
    read_phi(Doctor, PHI_Record, _Purpose, _EventID),  
    has_role(Doctor, 'doctor'),  
    owns_phi_record(Patient, PHI_Record),  
    \+ is_doctor_of(Doctor, Patient).
```

### Step 3: Conduct Expert Verification and Finalization

A human expert reviews the candidate rule for correctness, consistency, and safety.

- **Logical Correctness:** The expert verifies that the logic correctly identifies a violation when a doctor is \*not\* the designated physician (“`\+ is_doctor_of`”). This accurately reflects the “if and only if” nature of the policy text.
- **Schema Consistency:** The expert confirms that all predicates (`'read_phi'`, `'has_role'`, etc.) are consistent with the established system schema, preventing ambiguity.
- **Error Mitigation:** The expert checks for potential LLM hallucinations or misinterpretations. In this case, the model correctly identified all necessary conditions without adding extraneous or incorrect logic.

The rule is confirmed as correct and is integrated into the final policy corpus. This same rigorous process is applied to all other policy segments.

### Structured Prompt Example

#### INSTRUCTION:

Translate the following policy text into a declarative Prolog rule. The rule should define a **violation** that occurs when a required constraint is not met after a trigger event.

#### SCHEMA:

- Predicates: `read_phi(Principal, Resource, Purpose, EventID),`  
`has_role(Principal, Role),`  
`owns_phi_record(Patient, Resource),`  
`is_doctor_of(Doctor, Patient), ...`
- Rule Head: `violation(RuleName, Subject, Object).`
- Use '`\+`' for negation.

#### FEW-SHOT EXAMPLE:

Text: "A patient's record shall not be processed for a purpose if they have not given consent."

Rule:

```
violation('gdpr_art18_restriction', P, R) :-  
    read_phi(P, R, Purpose, _),  
    owns_phi_record(Patient, R),  
    \+ has_unrestricted_status(Patient, Purpose).
```

#### POLICY TEXT TO TRANSLATE:

"A principal acting in the role of a 'doctor' is only authorized to read a Patient Health Information (PHI) record if they are the designated physician for the patient who owns that record."

## B Formal Policy Examples

This appendix demonstrates the formalization of key compliance obligations from GDPR and HIPAA into the declarative Prolog rules used by our ACE auditor. Each rule defines the specific conditions that constitute a violation.

**Rule 1: HIPAA Authorization Control (hipaa\_auth)** A violation occurs if a principal with a 'doctor' role reads a PHI record belonging to a patient to whom they are not formally assigned. This enforces the core principle of authorized access.

```
violation('hipaa_auth', Doctor, PHI_Record) :-  
    read_phi(Doctor, PHI_Record, _Purpose, _EventID),
```

```

has_role(Doctor, 'doctor'),
owns_phi_record(Patient, PHI_Record),
\+ is_doctor_of(Doctor, Patient).

```

**Rule 2: HIPAA Minimum Necessary (hipaa\_min\_necessary)** This rule enforces that a principal may only access the specific types of data (attributes) necessary for their role. A violation is triggered if a principal reads a record containing an attribute that their assigned role is not permitted to access.

```

violation('hipaa_min_necessary', Principal, PHI_Record) :-
    read_phi(Principal, PHI_Record, _Purpose, _EventID),
    has_role(Principal, Role),
    read_attribute(Principal, PHI_Record, Attribute),
    \+ role_can_access_type(Role, Attribute).

```

**Rule 3: GDPR Art. 18, Restriction of Processing (gdpr\_art18\_restriction)** A violation occurs if a patient's record is processed for a specific purpose for which the patient has not given unrestricted consent. This enforces the data subject's right to control how their data is used.

```

violation('gdpr_art18_restriction', Principal, PHI_Record) :-
    read_phi(Principal, PHI_Record, Purpose, _EventID),
    owns_phi_record(Patient, PHI_Record),
    \+ has_unrestricted_status(Patient, Purpose).

```

**Rule 4: GDPR Art. 17, Right to Erasure (gdpr\_art17\_erasure)** This rule enforces the "right to be forgotten." A violation is detected if a patient's deactivation request remains unfulfilled for more than 30 days. The violation is attributed to the patient's assigned doctor, who is responsible for actioning the request.

```

violation('gdpr_art17_erasure', Doctor, RequestID) :-
    request_deactivation(Patient, RequestID, RequestDate),
    is_doctor_of(Doctor, Patient),
    current_date(Today),
    days_since(RequestDate, Today, Days),
    Days > 30,
    \+ deactivation_fulfilled(RequestID).

```

**Rule 5: GDPR Art. 15, Right of Access (gdpr\_art15\_access)** This rule upholds the patient's right to access their data in a timely manner. A violation occurs if a patient's formal request for their data is not fulfilled within the 30-day statutory limit. The patient's doctor is held responsible for this failure.

```

violation('gdpr_art15_access', Doctor, RequestID) :-
    request_access(Patient, _PHI_Record, RequestID, RequestDate),
    is_doctor_of(Doctor, Patient),
    current_date(Today),
    days_since(RequestDate, Today, Days),
    Days > 30,
    \+ request_fulfilled(RequestID).

```

## C System Parameters

To demonstrate the adaptability of the ACE framework to diverse organizational risk profiles, Tab. 3 provides a list of the system’s tunable parameters. These configuration options allow administrators to calibrate the scoring model’s sensitivity, dimension prioritization, and temporal dynamics.

Table 3: Comprehensive List of System Parameters

Parameter	Description	Impact on Reputation Computation
Time Window ( $W$ )	The discrete time period (e.g., weekly, monthly) over which violations are grouped and assessed.	Determines the update frequency of the score. A shorter window provides near real-time volatility; a longer window smoothes out behavioral spikes.
Rule Criticality ( $C$ )	A pre-assigned weight $C \in [0, 1]$ reflecting the intrinsic importance of a specific compliance rule.	Prioritizes high-risk obligations. A high $C$ (e.g., 0.95 for GDPR Erasure) ensures that a single violation significantly degrades the score, whereas low $C$ violations have minimal impact.
Scaling Factor ( $\alpha$ )	A scaling parameter unique to each dimension ( $\alpha_V, \alpha_T, \alpha_B$ ) used in the normalization function.	Controls the system’s sensitivity to violation magnitude. A small $\alpha$ creates a "strict" system where even minor violations result in high severity scores. A large $\alpha$ creates a "lenient" system requiring massive violations to trigger severe penalties.
Dimension Weights ( $w$ )	The relative weights ( $w_V, w_T, w_B$ ) assigned to Volume, Duration, and Breadth in the geometric mean calculation.	Tailors the score to organizational priorities. For example, increasing $w_V$ makes the score highly sensitive to massive data leaks, while increasing $w_T$ penalizes persistent, long-term non-compliance more heavily.
Decay Constant ( $\lambda$ )	A time-decay factor determining how quickly historical penalties are reduced over time.	Controls the "path to redemption." A high $\lambda$ allows the score to recover quickly after violations cease (high forgiveness). A low $\lambda$ causes penalties to linger, requiring a long period of compliant behavior to restore the score.