

SENTINEL: Evasion-Aware Trust Scoring, Deception, and Cryptographic Write Authorization for Industrial IoT

Mohammad Salman Khan^{1*}, Hammad Chughtai², Halil Ibrahim Ozkul¹ and Burak Yilmaz¹

¹Department of Software Engineering, Konya Technical University, Konya, Turkey

²Department of Computer Science (Cyber Security), University of Bradford, Bradford, United Kingdom

*Corresponding Author

Mohammad Salman Khan, Department of Software Engineering, Konya Technical University, Konya, Turkey.

Submitted: 2026, Jul 27; **Accepted:** 2026, Oct 19; **Published:** 2026, Oct 29

Citation: Khan, M. S., Chughtai, H., Ozkul, H. I., Yilmaz, B. (2026). SENTINEL: Evasion-Aware Trust Scoring, Deception, and Cryptographic Write Authorization for Industrial IoT. *Adv Mach Lear Art Inte*, 7(4), 01-18.

Abstract

Industrial Internet of Things (IIoT) environments require security controls that can distinguish legitimate operational changes from credential misuse, gradual process manipulation, and unauthorized command execution without violating realtime and availability constraints. Detection alone is insufficient when a compromised network-facing service can still issue writes to operational technology. This paper presents SENTINEL, an architecture that separates a public command broker from a protected trust decision point, routes uncertain sessions to a session-isolated deception environment, and requires a shortlived, HMAC-authenticated write capability for every live actuator update. A windowed trust scorer evaluates 152 statistical features across identity, temporal cadence, role authorization, and cross-sensor process consistency. Across five independent runs, each using a newly generated synthetic corpus of 192,000 commands organized into 4,000 sessions, SENTINEL achieved $98.40\% \pm 0.45\%$ accuracy, $97.00\% \pm 0.92\%$ macro-F1, and $99.76\% \pm 0.33\%$ simulated decision-path containment. In an end-to-end Modbus/TCP software testbed, 5,000 attack trials spanning role violation, replay, payload modification, slow drift, and broker retargeting caused no unauthorized modification of the protected OpenPLC register. Valid protected writes achieved a median end-to-end latency of 17.52 ms (P95: 18.56 ms, P99: 18.73 ms), adding 4.06 ms paired median overhead over the unprotected Modbus baseline. These results demonstrate the architectural feasibility of combining continuous behavioral scoring, active deception, and short-lived write authorization to limit unauthorized process manipulation following compromise of a network-facing component.

Keywords: Industrial IoT, Operational Technology, Trust Scoring, Cyber Deception, Honeypot, Command Authorization, Replay Prevention, Insider Threat, Evasion Attack

1. Introduction

Operational technology (OT) increasingly depends on interconnected sensors, programmable logic controllers (PLCs), remote engineering workstations, gateways, and cloud or enterprise services. This convergence improves observability and automation but exposes physical processes to software and identity compromise. OT defenses must therefore protect not only information assets, but also actions that can change a physical process. NIST emphasizes that OT security controls must preserve safety, reliability, and performance requirements that differ from conventional enterprise systems [1]. A recurring weakness in industrial architectures is that a successfully authenticated session is treated as trustworthy for the remainder of its interaction. Stolen

credentials, a malicious insider, or a compromised network-facing service can then issue commands that are individually valid but collectively unsafe. Slow manipulation is especially difficult because an adversary can remain within static ranges and spread a process change across many commands. Prior work has shown that stealthy industrial attacks can trade impact against detectability and can evade monitors that consider only isolated observations [2].

Three families of defenses address parts of this problem. Network intrusion detection systems, such as Kitsune, model packet-level or channel-level behavior [3]. Multivariate timeseries methods capture temporal and cross-sensor relationships [4-6]. Industrial

honeypots, including Conpot, provide deception and intelligence collection [7,8]. These approaches are valuable, but they usually operate in a detect-and-alert mode. They do not necessarily ensure that every physical write is authorized by an isolated policy decision point, and they generally do not use deception as an intermediate response to uncertainty.

This paper presents SENTINEL, a command-path architecture that combines four controls. First, a public broker cannot independently authorize a write. Second, a protected trust decision point evaluates policy and recent behavioral context. Third, uncertain sessions are transparently redirected to a session-isolated decoy state instead of being immediately blocked. Fourth, approved writes require a short-lived capability token that is bound to the exact worker, role, target, operation, value, nonce, and policy version. The design follows the zerotrust principle that access decisions should be made for the protected resource rather than inferred from network location [9].

The contributions are:

- A separation of the network-facing broker, trust decision point, deception environment, and OT enforcement point, with explicit trust boundaries and failure assumptions.
- A command-window trust scorer that combines identity, authorization, timing, replay, and process-context evidence and maps risk into *approve*, *deceive*, or *block* outcomes.
- A target-bound HMAC capability and one-time nonce protocol that prevents payload substitution, stale-token use, replay, and cross-device token reuse at an OT gateway.
- A reproducible session-level simulation study with a shifted held-out test domain, comparative baselines, feature ablations, token misuse tests, and CPU latency measurements.
- An empirical end-to-end Modbus/TCP software-testbed evaluation connecting Model A, Model B, an enforcement gateway, OpenPLC Runtime, and a simulated water-tank process.

The paper deliberately separates three claims that are often conflated in IIoT security research. *Detection* concerns whether a command sequence is recognized as suspicious; *containment* concerns whether uncertain or malicious activity is prevented from reaching the physical process; and *authorization integrity* concerns whether a network-facing compromise can substitute, replay, or redirect an otherwise valid write. This separation allows a system to remain useful even when the classifier is uncertain: a session can be contained in deception without claiming perfect attack attribution.

The remainder of the paper is organized as follows. Section II reviews temporal detection, industrial deception, and resourcecentric authorization. Section III defines the assets, design goals, adversary capabilities, and safety assumptions. Sections IV and V present the architecture, algorithms, implementation choices, and deployment patterns. Sections VI and VII describe the evaluation and testbed results. Sections VIII and IX discuss

security implications, threats to validity, and directions for broader physical-process and industrial deployment validation.

2. Background and Related Work

2.1. Industrial Detection and Datasets

Public testbeds and datasets have enabled repeatable ICS security research. SWaT provides a six-stage water-treatment testbed and associated attack data; TON IoT provides telemetry and network data for IoT and IIoT intrusion detection [10-12]. These resources are important, but their recorded variables do not directly represent all of the command-authorization fields required by SENTINEL, such as worker role, nonce freshness, request target binding, and session-specific deception state. The present study therefore uses a transparent synthetic command generator and treats real-plant validation as future work rather than presenting synthetic results as field performance.

Kitsune demonstrates efficient online network anomaly detection through an ensemble of autoencoders [3]. USAD, GDN, and TranAD model multivariate temporal behavior using adversarial autoencoders, learned graphs, and transformers, respectively [4-6]. These models focus primarily on anomaly detection. In contrast, SENTINEL's primary unit of control is an authenticated command request, and the output is an enforcement decision rather than an alarm alone.

2.2. Deception and Honeypots

Conpot emulates industrial protocols to collect intelligence on attackers targeting ICS services [7]. Surveys show that industrial honeypots can support threat intelligence and defensive experimentation, but also face realism and fingerprinting limitations [8]. Recent work demonstrates that observable network characteristics can reveal ICS honeypots, reinforcing the need to describe deception fidelity conservatively rather than claim indistinguishability [13].

Shadow honeypots combine anomaly detection with isolated execution and discard state changes caused by malicious requests [14]. SENTINEL adopts a related principle at the command-session level: uncertain commands are executed only against a decoy process state, while the live OT state remains unchanged. Unlike a general software shadow, the decoy is integrated with command authorization and an OT write enforcement point.

2.3. Resource-centric Authorization and Zero Trust

Zero-trust architecture does not imply that every request must traverse a cloud identity platform; rather, it requires that access to a protected resource be decided from authenticated identity, device, policy, and context instead of network location alone [9]. In an IIoT command path, the protected resource is not merely an API endpoint. It is the ability to mutate an actuator, PLC register, configuration parameter, or process setpoint. Consequently, an application-layer "allow" decision is insufficient unless a policy enforcement point adjacent to the OT resource verifies the decision.

SENTINEL implements this separation through a narrow write capability. The capability resembles a message authentication code over an immutable command context and is accepted only once during a short validity interval. HMAC provides integrity and source authentication when the decision point and enforcement point can share a protected key [15]. This mechanism is complementary to anomaly detection: hard authorization rejects commands that are invalid independent of behavior, while the scorer handles context-dependent cases that are individually legal but collectively suspicious.

2.4. Gap Addressed by This Work

The literature contains strong components for network anomaly detection, multivariate process monitoring, deception, and policy enforcement, but these components are usually evaluated

independently. A detector may report a suspicious session without controlling whether its next write reaches the plant; a honeypot may attract scanning traffic without being part of a live command decision; and a cryptographic gateway may verify integrity without detecting a legitimate credential being used gradually and maliciously. The research question addressed here is therefore architectural as well as predictive: can these controls be composed into one command lifecycle with measurable trade-offs among availability, uncertainty handling, and write prevention?

2.5. Positioning

Table 1 compares the principal capabilities relevant to this work. The table avoids subjective readiness scores; it records whether each work directly supports the listed function.

Approach	Command-level context	Temporal behavior	Enforced write gate	Active deception	OT/IIoT focus	Primary role
Kitsune [6] [3]	×	✓	×	×	△	Network anomaly Detection
USAD / GDN / TranAD [10]–[12] [4-6]	△	✓	×	×	△	Multivariate anomaly detection
Conpot [14] [7]	×	×	×	✓	✓	ICS honeypot and intelligence collection
Shadow honeypots [13] [14]	△	△	×	✓	×	Isolated validation of suspicious software requests
SENTINEL	✓	✓	✓	✓	✓	Command decision, deception, and write enforcement

✓ = directly supported; △ = partial or indirect support; × = not a native function.

Table 1: Capability Positioning of Representative Approaches

3. System and Threat Model

3.1. Design Objectives

The architecture is guided by six objectives. G1: mandatory mediation requires every remote write to pass through a verifier adjacent to the OT resource. G2: broker compromise tolerance requires Model B to remain unable to create a valid physical-write authorization by itself. G3: temporal context requires the decision point to evaluate a sequence rather than a single command. G4: uncertainty containment requires suspicious but inconclusive sessions to be isolated without immediately disrupting all legitimate activity. G5: bounded overhead requires the decision path to remain compatible with supervisory command rates. G6: auditable evidence requires decisions, token outcomes, and decoy interactions to be reconstructable without automatically poisoning the trusted baseline.

3.2. Formal System and State-Transition Model

We model the OT process as a discrete-time state-transition system over a continuous-valued plant state space:

$$M = (S, C, \delta, s_0) \quad (1)$$

where $S \subseteq \mathbb{R}^d$ denotes the continuous physical plant state space (pressures, flow rates, temperatures, and valve states), C is the domain of canonical normalized commands, $\delta : S \times C \rightarrow S$ is the underlying process dynamics transition function, and $s_0 \in S$ is the initial plant state.

Let $C_{\text{write}} \subset C$ represent the subset of state-mutating commands (setpoint updates or actuator toggles). In traditional, unprotected IIoT deployments, any issued command $c_i \in C_{\text{write}}$ directly triggers physical state evolution:

$$s_{t+1} = \delta(s_t, c_t) \quad (2)$$

SENTINEL decouples public command ingress from physical state mutation by introducing a dual-namespace state transformation $(\psi_{\text{live}}, \psi_{\text{decoy}}^{(k)})$:

$$s_{t+1}^{\text{live}} = \psi_{\text{live}}(s_t^{\text{live}}, c_t, T_t) \quad (3)$$

$$s_{t+1}^{\text{decoy},(k)} = \psi_{\text{decoy}}^{(k)}(s_t^{\text{decoy},(k)}, c_t) \quad (4)$$

where T_t is an HMAC-authenticated write capability, and $k \in \mathbb{N}$ denotes a session-isolated decoy namespace. The live state evolution ψ_{live} enforces the strict verification predicate:

$$\psi_{live}(s_t^{live}, c_t, T_t) = \begin{cases} \delta(s_t^{live}, c_t), & \text{if } \text{verifycap}(c_t, T_t, K_{epoch}) = \text{true} \\ s_t^{live}, & \text{otherwise} \end{cases} \quad (5)$$

Intuitive Interpretation: Physical plant actuators and setpoint registers update if and only if a fresh cryptographic capability token T_t is verified at the OT gateway. If an unverified or suspicious command arrives, the live physical process state remains locked (s_t^{live}), while the command is safely redirected to the session-isolated deception layer without causing physical plant disruption.

3.3. Protected Assets and Security Invariants

The protected assets are the live actuator/setpoint state, worker authorization context, session decision state, forensic evidence, identity-signing material, and capability-authentication key material. The central invariant is that live OT state changes only after the enforcement point verifies a fresh capability whose authenticated fields exactly match the pending command. Identity is signed independently of Model B; uncertainty is isolated in a

session-specific decoy namespace; audit evidence is append-only; and the broker never possesses the capability HMAC key.

3.4. System Entities

The architecture contains five security-relevant entities, as illustrated in Fig. 1:

- Ingress identity service. Authenticates the worker and device, validates request freshness, and produces a signed identity context.
- Model B, public command broker. Parses protocol messages, normalizes commands, and maintains session transport state. It does not make final trust decisions.
- Model A, trust decision point. Applies hard policy constraints, computes a behavioral score, chooses an action, and generates HMAC-authenticated write capabilities.
- Deception service. Maintains a session-isolated decoy state and returns protocol-consistent acknowledgments without changing the live process.
- OT enforcement point. Validates the capability, nonce, target, role, value, and expiration before forwarding a write to a PLC, actuator, or legacy gateway.

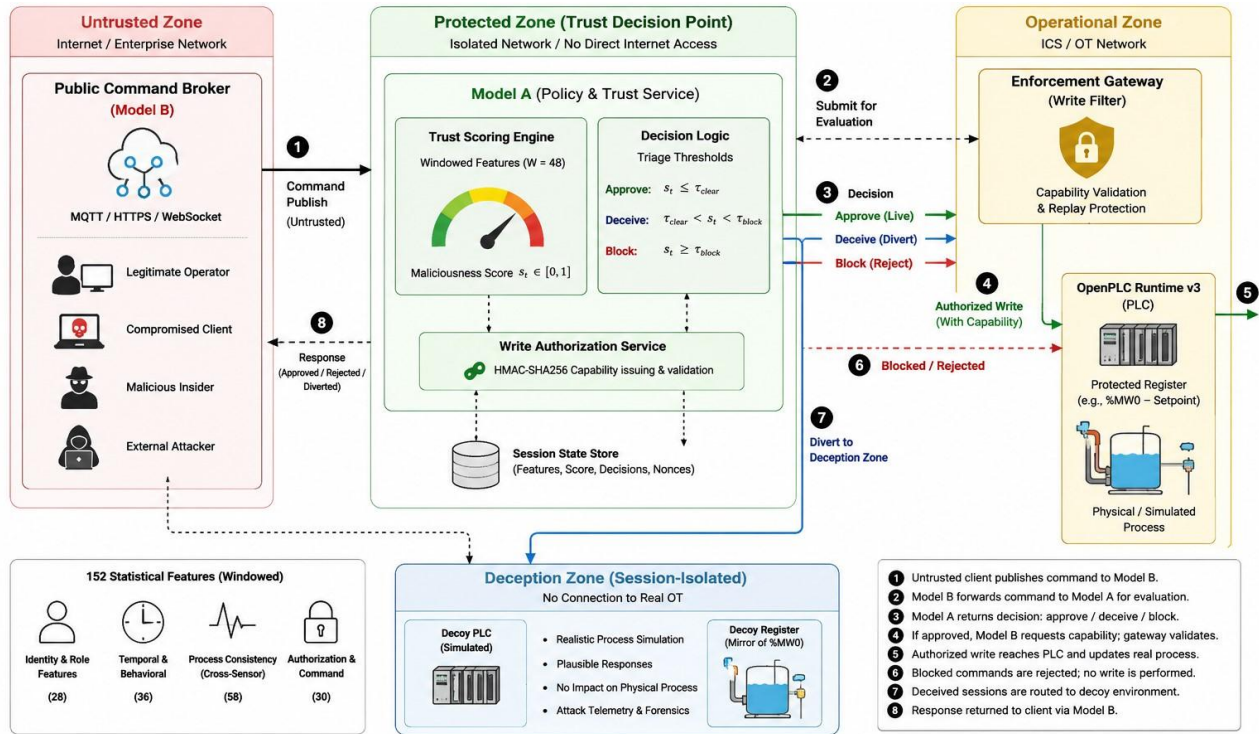


Figure 1: End-to-end SENTINEL command mediation architecture showing the Untrusted Zone with public broker (Model B), Protected Zone trust decision point (Model A) with 152 windowed statistical features ($W=48$) and HMAC-SHA256 capability issuance, Operational Zone write enforcement gateway and OpenPLC controller, session-isolated Deception Zone, and numbered 8-step command lifecycle

3.5. Adversary Capabilities and threat Model

We consider five operational attack classes and one architectural compromise. An insider may issue legal commands whose cumulative effect is unsafe; stolen credentials may be used from an unusual origin or cadence; a slow-evasion adversary may spread low-amplitude changes over time; an authenticated worker

may exceed role permissions; and a replay attacker may repeat, retarget, or modify captured commands. We also allow arbitrary code execution in Model B, including payload alteration and reordering.

Assuming the HMAC key remains secret and canonical serialization is unambiguous; an adversary cannot generate a valid capability for a new command without forging HMACSHA256. Uniform 128-bit nonces and atomic single-use consumption provide replay resistance and make accidental nonce collision (approximately $q_{\text{cap}}(q_{\text{cap}} - 1)/2^{129}$) negligible. Under the stated key-separation assumptions, Model B alone therefore cannot authorize a new or modified live write.

3.6. Trusted Computing Base and Exclusions

The trusted computing base contains Model A, the capability HMAC key, the OT enforcement point, and the identitysigning service. Physical tampering, compromise of Model A, compromise of the enforcement gateway, and denial-of-service against the control channel are outside the experimentally evaluated threat model. The behavioral scorer cannot guarantee detection of a perfectly in-distribution attack that produces no measurable identity, temporal, authorization, or processcontext deviation. These assumptions follow the adversarialML distinction between attacker knowledge, capability, and lifecycle access [16].

3.7. Safety and Failure Behavior

A security gateway must not replace local safety interlocks. If Model A is unavailable, remote writes fail closed, while local PLC safety logic, emergency shutdown, and manually authorized maintenance procedures remain independent. Reads may continue through a degraded path when organizational policy permits. A maintenance bypass, when required, must be explicit, time-bounded, physically or independently approved, and logged; it must not be implemented as a silent fail-open condition.

The reference prototype evaluates command integrity and authorization; it does not claim to certify process safety. A production deployment must define the safe state for each actuator, the maximum tolerated decision delay, clock-skew bounds, gateway restart behavior, and what happens to unused capabilities when a policy or key version changes.

4. Sentinel Design

4.1. End-to-End Command Lifecycle

A command passes through eight operational steps grouped into six logical stages, matching the numbered lifecycle in Fig. 1. First, an ingress identity service authenticates the worker and issues a signed context I_i . Second, Model B parses protocol inputs and publishes canonical command ct to Model A. Third, Model A evaluates hard constraints and windowed trust scores (s_i). Fourth, if approved ($s_i < \tau_{\text{clear}}$), Model A issues capability T_i . Fifth, the OT enforcement gateway verifies T_i and updates the OpenPLC register. Sixth, commands failing hard checks or exceeding τ_{block} are blocked. Seventh, suspicious sessions ($\tau_{\text{clear}} \leq s_i < \tau_{\text{block}}$) are diverted to deception. Eighth, execution status is returned to the client via Model B. No response from Model B alone is sufficient to mutate the live physical state.

4.2. Command Context and Hard Constraints

A normalized command at time t is represented as

$$c_t = (u, r, d, o, v, n, t_i, t_e, p) \quad (6)$$

where u is the worker identity, r the role, d the target device or register, o the operation, v the value, n a nonce, t_i and t_e the issue and expiry times, and p the policy version. Before behavioral scoring, Model A rejects malformed requests, stale identity contexts, role violations, unsafe values, and operations outside the target policy. Hard violations are never routed to the live process.

The policy evaluation order is intentional. Syntax and freshness checks occur before any expensive model operation. Role and target authorization then establish whether the operation is categorically permitted. A device-specific safe envelope rejects values that are unacceptable regardless of behavioral history. Only a command that is individually admissible proceeds to contextual scoring. This arrangement avoids using a probabilistic model to override deterministic safety or authorization rules.

During session startup, before 48 commands are available, Model A pads the window by repeating the initial command state. As new observations arrive, padded entries are progressively evicted. Deterministic authorization and safety constraints remain active throughout startup and are never relaxed.

4.3. Command-Window Feature Construction

For commands that satisfy hard constraints, Model A maintains a sliding session window of the $W = 48$ most recent observations for each worker session $S_i = (c_{t-W+1}, \dots, c_t)$. During startup, before W commands accumulate, the window is padded by repeating the initial state; as new commands arrive, padded entries are evicted. From this matrix, Model A constructs a 152-dimensional summary vector $\phi(S_i)$ spanning eight temporal statistics across 19 base command features.

For each feature channel $j \in \{1, \dots, 19\}$, the extraction operator $\Phi_j(S_i) \in \mathbb{R}^8$ evaluates eight temporal summary statistics (mean μ_j , standard deviation σ_j , minimum, maximum, end-to-start difference $\Delta_j = c_{t,j} - c_{t-W+1,j}$, linear slope ∇_j , first quartile $Q1_j$, and third quartile $Q3_j$). The linear slope ∇_j is calculated via ordinary least-squares regression:

$$\nabla_j = \frac{\sum_{k=1}^W (k - \bar{k})(c_{t-W+k,j} - \mu_j)}{\sum_{k=1}^W (k - \bar{k})^2}, \bar{k} = \frac{W + 1}{2} \quad (7)$$

Concatenating these operators across all 19 feature channels yields the complete session summary vector:

$$\Phi_j(S_t) = [\mu_j \sigma_j \min_j \max_j \Delta_j \nabla_j Q1_j Q3_j]^T \quad (8)$$

$$\phi(S_t) = \text{Contact}(\Phi_1(S_t), \Phi_2(S_t), \dots, \Phi_{19}(S_t)) \in \mathbb{R}^{152} \quad (9)$$

Group	Features	Security interpretation
Operation and target	read/write/setpoint indicators; pump/valve/heater indicators	Distinguishes ordinary monitoring from state changing operations and captures device-specific patterns
Process Context	normalized value, normalized delta, cross-sensor residual, safe-envelope flag	Detects cumulative drift and inconsistency between a requested value and related process observations
Identity and authorization	role-permission flag, origin-trust score	Identifies commands that conflict with the role or appear from an unusual authenticated context
Freshness and replay	nonce-freshness flag, repeated-command similarity	Captures stale or duplicated command material and replay-like sequences
Timing and workload	log inter-arrival time, rate deviation, time-of-day sine/cosine	Models cadence, bursts, shift changes, and unusual access times without treating one time value as categorically malicious
Session Position	normalized session progress	Allows the scorer to distinguish benign startup/maintenance transients from evidence that emerges late in a session

Table 2: Groups the Command-Level Evidence by Security Purpose

Within the process-context feature group, cross-sensor consistency is continuously tracked via an Exponentially Weighted Moving Average (EWMA) residual $\hat{r}_t$, while operational diversity is measured using Shannon entropy $H(O_t)$:

$$\hat{r}_t = \beta \hat{r}_{t-1} + (1 - \beta) |v_t - f_{phys}(x_t)| \quad (10)$$

$$H(O_t) = - \sum_{o \in O} p(o) \log_2 p(o) \quad (11)$$

where $f_{phys}(x_t)$ estimates expected setpoint response from correlated sensor readings x_t , $\beta \in (0,1)$ is a smoothing factor, and $p(o)$ is the empirical probability of operation o within window St .

Physical Feature Interpretation: Each aggregated feature component captures a distinct physical attack footprint:

- **Linear Slope (∇):** Detects rapid or stealthy incremental setpoint manipulation over time.
- **EWMA Residual ($\hat{r}_t$):** Quantifies physical inconsistency between requested setpoints v_t and expected plant telemetry $f_{phys}(x_t)$.
- **Shannon Entropy ($H(O_t)$):** Measures operational diversity, spotting automated polling loops or unnatural command repetition.

Evasion awareness arises from training and evaluating the scorer on gradual, low-amplitude manipulation sequences, together with temporal slope, cumulative-delta, cadence, and cross-sensor residual features; it does not imply certified adversarial robustness. Startup-window behavior is treated as a limitation because repeating the initial command state during startup produces feature distributions that differ from mature windows.

The deployed reference scorer is a 160-tree random forest with maximum depth 15, minimum leaf size 3, square root feature

sampling, and balanced subsampling. A tree ensemble was selected over a recurrent model because it provided stronger shifted-test performance and interpretable feature importance in the reference study.

Let $P(y = k | \emptyset(s_t))$ be the class probability for the current session summary $\emptyset(s_t)$. The maliciousness score is

$$s_t = 1 - P(y = normal | \emptyset(s_t)) \quad (12)$$

The response is

$$D(s_t) \begin{cases} \text{approve,} & s_t < \tau_{clear} \\ \text{deceive,} & \tau_{clear} \leq s_t < \tau_{block} \\ \text{block,} & s_t \geq \tau_{block} \end{cases} \quad (13)$$

Thresholds are selected on the validation set under a constraint that immediate false blocks remain below 2%. Candidate pairs are ranked using

$$U = 1.6C_a + 0.8A_b + 0.25B_a - 3.2B_b \quad (14)$$

where C_a is attack containment, A_b is benign approval, B_a is immediate attack blocking, and B_b is benign immediate blocking. Utility coefficients and candidate threshold ranges were specified on the validation set prior to evaluation on the held-out test set; the test partition was not used for weight tuning or threshold selection. The selected values are $\tau_{clear} = 0.45$ and $\tau_{block} = 0.65$.

4.4. Deception State Algebra and Session Transitions

The deception service allocates a state namespace per suspicious session. On the first diversion at command index t_{div} , the decoy state is initialized via a snapshot operator:

$$s_0^{decoy,k} = \text{Sanitize}(s_{t_{div}}^{live}) \quad (15)$$

where Sanitize decouples the decoy state from live actuator registers, removes administrative credentials and historian metadata, and resets physical sensor accumulators while retaining non-sensitive setpoints and process telemetry. Subsequent reads and writes from session k are resolved against that isolated namespace:

$$s_{t+1}^{decoy,k} = g_{plant}(s_t^{decoy,k}, c_t) \quad (16)$$

where g_{plant} simulates protocol-consistent state updates without updating s^{live} . The differential process divergence between live and decoy states is tracked as:

$$\Delta_{decoy}(t) = \| s_t^{live} - s_t^{decoy,k} \|_2 \quad (17)$$

4.5. Deception Mechanics and Timing Equalization

When a session produces score $S_t \in [\tau_{clear}, \tau_{block}]$, Model A routes the connection to a session-isolated deception worker.

The worker maintains decoy state $\psi_{decoy}^{(k)}$ initialized from benign operational values and returns protocol-compliant responses.

To reduce simple timing distinguishability between the live and decoy paths, the deception gateway samples an additional delay from a non-negative empirical latency distribution collected from the live OT gateway:

$$t_{resp} = t_{proc} + \max(0, \tilde{L}_{live} - t_{proc}) \quad (18)$$

where $\tilde{L}_{live}$ is sampled from measured live-path latency traces

and t_{proc} is the decoy processing time. This mechanism does not guarantee resistance to advanced timing analysis, but is intended to reduce obvious timing discrepancies while maintaining session-consistent responses.

Operational Mechanics: Suspicious sessions receive realistic protocol status responses and consistent read-after-write values from g_{plant} , combined with calibrated round-trip timing jitter. This is intended to preserve protocol and read-after-write consistency while reducing obvious timing differences between the live and decoy paths. Decoy responses do not alter the live physical-process state.

The session remains in deception until one of four events occurs: the score crosses τ_{block} , the session expires, an operator explicitly terminates it, or a de-escalation policy confirms sufficient benign evidence. The reference experiment evaluates diversion and block decisions at the session level; it does not claim a complete industrial process simulator. The decoy is therefore not described as indistinguishable from a real plant. Its purpose is controlled observation and containment. Raw decoy interactions are stored as forensic evidence and are not automatically added to trusted training data, avoiding a direct poisoning path.

4.6. Capability Generation and Structure

For an approved command c_t , Model A generates a shortlived write capability:

$$K_{epoch} = \text{KDF}(K_{master}, epoch \parallel p) \quad (19a)$$

$$T_t = \text{HMAC}_{K_{epoch}}(\text{Ser}(u, r, d, o, v, n, t_i, t_e, p)) \quad (19b)$$

The reference lifetime is $\Delta t_{expire} = t_e - t_i = 800$ ms. The enforcement point recomputes the HMAC, verifies the exact

Algorithm 1 SENTINEL command decision and capability issuance

Require: command c_t , signed identity context I_t , session history S_t , master key K_{master}

- 1: if I_t is invalid or $\text{CheckFreshness}(I_t) = \text{False}$ then
- 2: return Block and append audit alert
- 3: end if
- 4: if c_t violates role r , target d , or safe envelope $[v_{min}, v_{max}]$ then
- 5: return Block and log policy violation
- 6: end if
- 7: append c_t and feature vector to session window S_t

```

8: extract 152-dimensional summary vector  $\phi(S_t)$ 
9:  $s_t \leftarrow 1 - P(y = \text{normal} \mid \phi(S_t))$  using the random-forest ensemble
10: if  $s_t < \tau_{\text{clear}}$  then
11:    $K_{\text{epoch}} \leftarrow \text{KDF}(K_{\text{master}}, \text{epoch} \parallel p)$ 
12:    $T \leftarrow \text{HMAC}_{K_{\text{epoch}}}(u \parallel r \parallel d \parallel o \parallel v \parallel n \parallel t_i \parallel t_e \parallel p)$ 
13:   return Approve with capability  $T$ 
14: else if  $s_t < \tau_{\text{block}}$  then
15:    $S_{\text{decoy}}^{(k)} \leftarrow \text{InitDecoySession}(k, c_t)$ 
16:   return Deceive via session-isolated decoy state
17: else
18:   return Block, persist evidence record, and alert SOC
19: end if

```

target and value, checks role and safe-envelope constraints, validates the policy version and time interval, and rejects any previously used nonce. A token for one device, value, or operation cannot authorize another. HMAC is used in the reference prototype for low overhead; production deployments may use asymmetric capabilities or hardware-protected keys when organizational key distribution requires them.

4.7. Security Properties

Under the stated trust assumptions, the command path provides four bounded properties. P1: exact-command integrity. Modification of any bound field causes the enforcement point's recomputed HMAC to differ. P2: single-use freshness. A captured capability cannot be accepted after expiry or after its nonce has been

atomically consumed. P3: broker nonauthority. Model B cannot independently authorize a write because it lacks the capability HMAC key and cannot produce an independently signed ingress identity context. P4: uncertainwrite isolation. A deceive decision resolves the command against a session-specific namespace and therefore does not mutate live OT state.

These are implementation-level properties, not a proof of whole-system safety. They depend on key confidentiality, correct canonical serialization, atomic nonce handling, enforcement-point integrity, and the assumption that all remote writes are mediated. The design does not prevent a local physical override or a compromised PLC from altering its own state.

Algorithm 2 OT enforcement-point verification

Require: pending command c , capability T , system clock t , nonce cache N_{used} , epoch key K_{epoch}

```

1: if  $T = \emptyset$  or key identifier is unknown then
2:   return Reject("Missing token")
3: end if
4: if  $t < t_i - \Delta t_{\text{skew}}$  or  $t > t_e$  then
5:   return Reject("Expired/Stale token")
6: end if

```

```

7: if  $n \in N_{\text{used}}$  or policy version  $p$  is inactive then
8:   return Reject("Replay or Inactive Policy")
9: end if
10: if  $r, d, o, v$  violates local static safety bounds then
11:   return Reject("Local Safety Violation")
12: end if
13:  $T' \leftarrow \text{HMAC}_{K_{\text{epoch}}}(u \parallel r \parallel d \parallel o \parallel v \parallel n \parallel t_i \parallel t_e \parallel p)$ 
14: if CT-Compare( $T', T$ )  $\neq$  Equal then
15:   return Reject("Capability Authentication Mismatch")
16: end if
17: AtomicAdd( $N_{\text{used}}, n$ )
18: ForwardWriteToPLC( $d, o, v$ )
19: return Accept and record audit log

```

4.8. Computational and Storage Complexity

Let W be the session-window length, F the number of command features, and M the number of trees. Updating the fixed-window statistics is $O(F)$ with incremental aggregates or $O(WF)$ when recomputed as in the reference script. Tree inference is $O(MD)$ for average tree depth D . The reference values $W = 48$, $F = 19$, $M = 160$, and $D \leq 15$ keep the path small enough for supervisory traffic. Per-session state consists of the recent command window, the current decision mode, decoy values when applicable, and a bounded set of unexpired nonces.

5. Reference Implementation and Deployment

5.1. Software Components

The reproducibility package implements the synthetic generator, feature aggregation, model training, threshold selection, capability tests, and latency benchmark in Python. The implementation stores each complete session as a 48×19 tensor and exports the baseline-specific preprocessing objects, class map, and trained random-forest model in one versioned artifact. The token verifier uses canonical field ordering and constant-time MAC comparison. Random seeds, split rules, model hyperparameters, and generated result files are included so that the reported synthetic experiment can be reproduced.

A production implementation can separate these functions by language and host without changing the protocol. Model B may be an HTTP, MQTT, OPC UA, or Modbus-facing adapter; Model A may run as an isolated policy service; and the enforcement point may be embedded in a protocol gateway. The critical requirement is not a specific framework but preservation of the trust boundaries

and exact command binding.

5.2. Deployment Patterns

Two deployment patterns are considered. In a *native-verifier* deployment, a capable PLC, secure edge controller, or actuator firmware validates the capability directly. This minimizes the distance between the protected state and the enforcement logic but requires firmware support and secure key storage. In a *legacy-gateway* deployment, a dedicated gateway is the only routable writer to a legacy PLC. The gateway validates the capability and then emits the protocol-specific command. This is easier to integrate but expands the trusted computing base to include the gateway and the network segment between gateway and PLC.

5.3. Key, Nonce, and Policy Lifecycle

Capabilities include a key identifier and policy version so that rotation can occur without accepting commands under an ambiguous policy. During key rotation, the enforcement point may temporarily accept the current and immediately previous key identifiers, subject to their original short expirations. Nonces must be consumed atomically before forwarding the live write; otherwise concurrent duplicate submissions could race. A restart-safe implementation persists or reconstructs all still-valid nonces, or invalidates all outstanding capabilities by advancing the key or policy version.

Audit records should contain the canonical command hash, identity-context identifier, model version, score, selected action, capability identifier, verifier result, and monotonic event time. Sensitive worker identifiers can be pseudonymized in analytical exports

while remaining recoverable under controlled incident response procedures.

6. Experimental Methodology

6.1. Research Questions

The evaluation addresses five questions:

- RQ1: Can the scorer distinguish normal sessions from five command-level attack classes under a shifted test domain?
- RQ2: What is the operational trade-off among approval, deception, and immediate block thresholds?
- RQ3: Does the capability verifier reject common token and payload misuse cases?
- RQ4: Is the in-process decision path compatible with low millisecond command handling?
- RQ5: Does the complete SENTINEL command path prevent unauthorized Modbus/TCP writes and remain operationally feasible in an end-to-end software testbed?

6.2. Synthetic Command Corpus

The corpus contains 4,000 complete sessions and 192,000 commands. Sixty percent of sessions are normal; each of five attack classes contributes 8%. Attack sessions begin after a benign prefix and then introduce one of the following patterns:

- Insider drift: gradual setpoint change with increasing cross-sensor residual.
- Credential misuse: origin and cadence mismatch with mostly valid values.
- Slow evasion: low-amplitude cumulative drift that remains within a static safe envelope.
- Privilege abuse: intermittent writes outside the worker role.
- Replay: repeated commands with stale nonces and elevated similarity.

Normal sessions include remote-maintenance, process-transient, and repeated-polling episodes so that single anomalous features are not sufficient for classification. Each session has a randomized role, device type, starting value, cadence, start hour, origin-trust baseline, and read/write/setpoint distribution. Attack onset is sampled between commands 20 and 33 in the base generator, ensuring a benign prefix. Table 3 summarizes the evidence injected by each attack template. Values are clipped to a normalized range and static unsafe-envelope violations are deliberately rare in the drift classes; this prevents a simple range rule from solving the task.

Class	Generated behavior after onset	Primary evidence available to scorer	Why a static rule is insufficient
Insider drift	Incremental setpoint movement and gradually increasing process residual	Value slope, end-to-start change, residual trend and dispersion	Most individual values remain within the safe envelope and originate from a permitted worker
Credential misuse	Lower/more variable origin trust, cadence change, modest workload-rate increase	Origin-trust minimum/trend, inter-arrival statistics, rate deviation	Command syntax, role, and values are usually valid
Slow evasion	Smaller cumulative drift with oscillation and weak residual growth	Long-window value and residual slopes	Each update is intentionally below a conspicuous per-command threshold
Privilege abuse	Intermittent write/setpoint operations for which role permission is false	Role-permission minimum and variation, operation mix	Violations are sparse and can be surrounded by ordinary reads
Replay	Repeated operation/value templates, stale nonces, elevated similarity	Nonce freshness and repeated-command statistics	Replayed commands may contain otherwise safe and historically common values

Table 3: Synthetic Attack Templates and Discriminating Evidence

The split is stratified by complete session: 70% training, 15% validation, and 15% testing. No command window from one session appears in more than one split. The test set is further shifted by delaying attack evidence, mixing attack traces with independent benign profiles, and injecting mild maintenance-like features into

some normal sessions. This shift is intentionally modest: it tests whether the classifier survives overlap and later evidence, but it is not equivalent to an independently collected plant. Table 4 summarizes the split.

Class	Total	Train	Val.	Test
Normal	2400	1680	360	360
Insider drift	320	224	48	48
Credential misuse	320	224	48	48
Slow evasion	320	224	48	48
Privilege abuse	320	224	48	48
Replay	320	224	48	48
Total sessions	4000	2800	600	600

Table 4: Synthetic Session Corpus Distribution

6.3. Baselines and Hyperparameters

We compare the SENTINEL scorer against fixed rules, balanced multinomial logistic regression, and histogram gradient boosting. All learned models use the same 152-dimensional session summaries and the same session splits. Table 5 records the configuration used in the reproducibility package. The rules

baseline uses hand-written thresholds on role violations, origin trust, nonce freshness, repeated-command similarity, value drift, process residual, and rate deviation. It represents a plausible low-complexity alternative rather than a published system whose input representation differs from the command corpus.

Model	Configuration
Rules	Seven feature-derived threshold conditions; first matching attack label returned
Logistic regression	Multinomial, balanced class weights, $C=1.2$ maximum 700 iterations
Histogram boosting	180 iterations, learning rate 0.07, 25 maximum leaf nodes, $L_2=0.15$, balanced weights
SENTINEL random forest	160 trees, depth 15, minimum leaf 3, square root feature sampling, balanced subsampling

Table 5: Evaluated Model Configurations

6.4. Performance Metrics

Standard classification metrics—accuracy, macro-precision, macro-recall, macro-F1, balanced accuracy, and benign falsepositive rate (FPR)—are calculated on the held-out test set. Although the online scorer produces a decision after each command arrival, classification metrics are calculated from the terminal session prediction. We distinguish two levels of containment. Simulated decision-path containment is recorded in the synthetic evaluation when no malicious command is assigned live approval before the session is diverted or blocked. Physical containment is recorded in the executable Modbus/OpenPLC testbed only when register readback confirms that no unauthorized command modified protected live state. Precision and recall are macro-averaged across all six classes to prevent the larger normal class from dominating reported performance. Binary ROC-AUC and PRAUC treat all five attack classes as a single positive class to evaluate benign vs. malicious separation. To express test-set sampling uncertainty, we perform 2,000 nonparametric bootstrap resamples of the 600 held-out sessions and report percentile 95% confidence intervals for accuracy, macro-F1, FPR, and attack containment. These intervals do not capture generator uncertainty, model-development choices, or deployment shift; they quantify only resampling uncertainty within the fixed test corpus.

6.5. Capability and Latency Tests

The verifier is evaluated using a parameterized test suite containing 14,000 generated misuse cases (2,000 attempts for each of seven misuse patterns: missing token, modified value, expired token,

wrong target, replayed nonce, unauthorized role, and unsafe value). A valid control case is executed first. Each negative case changes one relevant condition while preserving the others, allowing the rejection reason to be attributed to the corresponding verification step.

Latency is measured on an AMD EPYC 9V74 virtual host with five vCPUs and 5.9 GiB RAM using Python 3.13 and scikit-learn 1.8. The scorer benchmark uses one model thread after warm-up; the shared preprocessing and scoring pipeline uses 1,000 invocations. HMAC issue and verification are measured separately. Measurements exclude network transport, queueing, database access, audit persistence, protocol translation, and PLC response time. The reported values are therefore computational microbenchmarks rather than end-to-end plant latency.

6.6. Reproducibility Protocol

The artifact records the random seed, generated class counts, complete-session split, trained model, feature names, threshold values, per-class metrics, ablations, token outcomes, latency samples, and figure-generation scripts. The artifact additionally contains the Modbus/TCP testbed implementation, OpenPLC configuration, process-register map, scenario definitions, raw per-trial latency and register-readback records, environment manifest, and scripts that regenerate the testbed tables. Rerunning the package regenerates the synthetic corpus and testbed figures rather than relying on manually entered tables.

7. Results

7.1. Classification Performance (Within-Generator Shifted Evaluation)

Table 6 reports the shifted-test results under within generator evaluation. Across 600 held-out shifted test sessions, the SENTINEL scorer achieves 98.50% accuracy, 97.40% macro-F1,

and a 0.28% FPR. Gradient boosting achieves 97.83% accuracy and 95.80% macro-F1, while the linear and rule baselines fail to separate several temporally overlapping classes. The high binary AUC values should be interpreted alongside the lower multiclass macro-F1: distinguishing attack from normal is easier than identifying the exact attack type.

Model	Acc.	Bal. Acc.	Macro-P	Macro-R	Macro-F1	FPR	ROC-AUC	PR-AUC
Rules only	60.17	17.01	27.21	17.01	13.60	0.00	—	—
Logistic regression	64.67	33.01	54.59	33.01	25.79	6.11	92.66	90.15
Gradient boosting	97.83	95.14	96.50	95.14	95.80	0.28	99.91	99.88
SENTINEL scorer	98.50	97.18	97.40	97.18	97.40	0.28	99.88	99.85

Table 6: Shifted-Test Classification Results (%)

Bootstrap intervals in Table 7 show that the headline result remains high under resampling of the representative seed (20260724), with macro-F1 showing expected sampling variation due to 48 test sessions per attack class.

Metric	Point	Lower	Upper
Accuracy (%)	98.50	97.50	99.33
Macro-F1 (%)	97.40	97.40	98.67
Benign FPR (%)	0.28	0.00	0.83
Simulated containment (%)	99.18	98.20	99.80

Table 7: Session-Bootstrap 95% Confidence Intervals (SEED 20260724)

Evaluating class-level recall on seed 20260724, replay sessions achieve 100.0% recall, credential misuse 97.9%, privilege abuse 97.9%, slow evasion 95.8%, and insider drift 91.7%. Figure 2 illustrates the multi-class confusion matrix on the shifted test set.

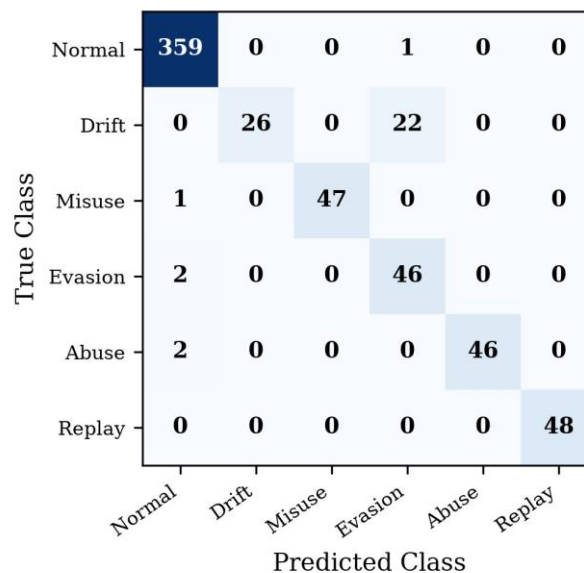


Figure 2: Shifted-test multi-class confusion matrix for the SENTINEL random forest scorer across the six evaluated session classes

The score distribution shows strong benign/attack separation, as illustrated in Fig. 3, with a small overlap near the clear threshold for credential-misuse, slow-evasion, and privilegeabuse sessions. This overlap motivates a deception band rather than an all-or-nothing classifier threshold.

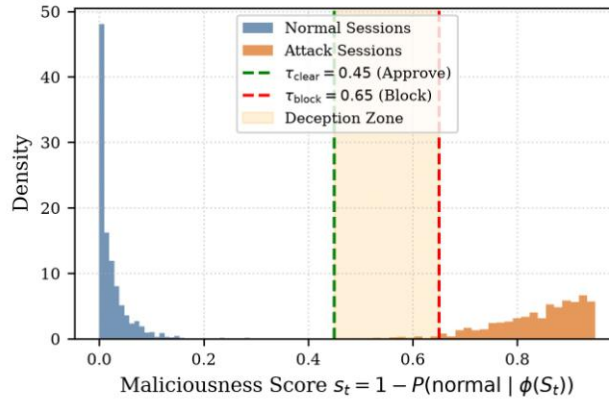


Figure 3: Distribution of maliciousness score s_t for normal and attack sessions showing the clear threshold ($\tau_{\text{clear}}=0.45$), block threshold ($\tau_{\text{block}}=0.65$), and the intermediate deception zone

7.2. Triage Behavior

At $\tau_{\text{clear}} = 0.45$ and $\tau_{\text{block}} = 0.65$, 99.17% of benign sessions are approved, 0.83% enter deception, and none are immediately blocked. Among attack sessions, 99.17% achieve simulated

decision-path containment: 95.00% are immediately blocked, 4.17% enter deception, and 0.83% are approved (Table 8). This demonstrates that the two-threshold policy exposes an operational trade-off that a single threshold would hide.

Class	Approve	Deceive	Block
Normal	99.17	0.83	0.00
Insider drift	0.00	8.33	91.67
Credential misuse	2.08	2.08	95.83
Slow evasion	2.08	8.33	89.58
Privilege abuse	0.00	2.08	97.92
Replay	0.00	0.00	100.00

Table 8: Decision Distribution by True Class (%) For SEED 20260724

Slow evasion produces the largest deception proportion, which is consistent with the intended role of the middle band: sessions with gradual evidence receive additional isolation and observation rather than immediate live access. Privilege abuse sessions were fully contained in the representative run: 97.92% were immediately blocked and 2.08% were redirected to deception. In production, the hard policy stage should still reject each disallowed operation independently of the session level score.

7.3. Feature ablation

Table 9 evaluates the contribution of feature groups on seed 20260724. Removing identity and command-context features reduces macro-F1 from 97.40% to 60.78%; removing process context features reduces it to 69.90%. Means alone retain 91.25% macro-F1, but static extrema alone perform poorly and raise the FPR to 7.80%. The result supports the central design claim that authorization and process evidence should be evaluated together over time.

Configuration	Features	Macro-F1 (%)	FPR (%)
Full model	152	97.40	0.28
No identity/context	120	60.78	4.44
No process context	120	69.90	4.72
Mean only	19	91.25	1.11
Static extrema only	38	72.40	7.80

Table 9: Feature Ablation on the Shifted Test Set (SEED 20260724)

The dominant features are the slope and dispersion of cross sensor residual, the trend and minimum of origin trust, role permission variation, repeated-command similarity, and nonce freshness. This distribution is consistent with the intended multi-source trust

design rather than dependence on a single static threshold.

7.4. Error Analysis

The principal attribution error is confusion between insider drift

and slow evasion. Both classes modify values gradually and increase the cross-sensor residual; their difference is primarily attack amplitude and onset profile. For containment, this confusion is benign because both labels map to a nonmoral score. For forensic attribution, however, the distinction is underdetermined from command summaries alone. A process aware residual model, plant-mode indicator, or graph relating dependent sensors would provide stronger evidence.

The small number of normal sessions near τ_{clear} contain maintenance-like transients deliberately mixed into the test set. This is operationally important: a low FPR measured only on stationary normal traffic can substantially underestimate disruption

during calibration, shift changes, or remote maintenance. Long-duration benign evaluation should therefore report false diversions and blocks per hour and stratify them by plant mode.

7.5. Capability Enforcement

All 14,000 unauthorized attempts were rejected, as summarized in Table 10. The verifier accepted a valid command and rejected every missing-token, modified-value, expiredtoken, wrong-device, replay, unauthorized-role, and unsafevalue attempt. These tests establish implementation correctness for the evaluated misuse cases; they do not prove resistance to key extraction or compromise of the trusted enforcement point.

Case	Attempts	Accepted
Missing token	2000	0
Modified value	2000	0
Expired token	2000	0
Wrong device	2000	0
Replay	2000	0
Unauthorized role	2000	0
Unsafe value	2000	0

Table 10: Capability Misuse Tests

7.6. Latency

On a five-vCPU AMD EPYC host, the complete preprocessing and scoring path has a median of 5.10 ms and a p99 of 7.87 ms over 1,000 warm invocations (Table 11). Capability generation and verification add 0.003 ms median. The computational

microbenchmark excludes network transport, protocol translation, audit persistence, and PLC response. End-to-end measurements incorporating these components are reported separately in the Modbus/TCP testbed evaluation.

Stage	Mean	P50	P95	P99
Scorer only	4.797	4.678	5.281	6.667
Full preprocessing + scoring	5.277	5.100	6.051	7.871
HMAC issue + verify	0.0030	0.0030	0.0040	0.0048

Table 11: In-Process Latency on Five Amd Epyc Vcpus (MS)

7.7. Multi-seed Pipeline Variance

To evaluate whether classifier performance is sensitive to stochastic dataset sampling or model initialization, the full pipeline—including synthetic session generation, feature

extraction, baseline-specific preprocessing, random forest training, and shifted-test evaluation—was executed across five independent random seeds (20260724 to 20260728). Table 12 records the per-seed breakdown.

Seed	Accuracy	Macro-F1	FPR	Simulated Containment
20260724	98.50	97.40	0.28	99.18
20260725	97.67	95.36	0.28	100.00
20260726	98.67	97.19	0.00	100.00
20260727	98.17	96.89	0.00	99.61
20260728	99.00	98.17	0.00	100.00
Mean ± Std	98.40 ± 0.45	97.00 ± 0.92	0.11 ± 0.14	99.76 ± 0.33

Table 12: Performance Across Five Independent Random Seeds (%)

Across all 5 seeds, SENTINEL achieves mean test accuracy of $98.40\% \pm 0.45\%$, macro-F1 of $97.00\% \pm 0.92\%$, benign false-positive rate of $0.11\% \pm 0.14\%$, and simulated decisionpath containment of $99.76\% \pm 0.33\%$. The observed variation is low across repeated runs of the same generator and pipeline, but does not measure robustness to independent data-generation assumptions. Feature scaling is retained for unified baseline evaluation (e.g., logistic regression) but does not alter the decision boundaries of tree-based models.

7.8. End-to-End Modbus/TCP Testbed Validation

To complement the synthetic within-generator evaluation, we deployed the complete SENTINEL command path in an end-to-end Modbus/TCP software testbed comprising an operator client, Model B broker, Model A policy service, enforcement gateway, and an OpenPLC runtime controlling a water-tank process simulator. The components communicated through active TCP sockets over a dedicated network interface, with host, software, and network parameters detailed in Table 13.

Component	Configuration / Specification
Host environment	5 vCPU AMD EPYC 9V74, 5.9 GiB RAM Linux 6.6
Operator client	Python 3.13 socket client, 1,000 trials per scenario
Broker (Model B)	Isolated protocol proxy, payload parsing, transport state
Policy service (Model A)	Contextual scorer, 48-command window, threshold policy
Enforcement gateway	Atomic 128-bit nonce cache, HMAC verifier socket sink
PLC Runtime	OpenPLC Runtime v3 simulation, Modbus/TCP port 502
Plant model	Water-tank process dynamics, holding register %MW0
Network transport	Dedicated IPv4 local socket transport, 1 Gbps interface

Table 13: MODBUS/TCP Testbed System Configuration

Baseline latency t_{baseline} was measured from client transmission of a Modbus write request to receipt of the corresponding OpenPLC socket response over 1,000 trials (100 warm-up requests discarded). Protected latency $t_{\text{protected}}$ was measured over the identical interval, incorporating broker parsing, 152feature extraction, Model A random-forest scoring, capability generation, enforcement-gateway verification, Modbus forwarding, register update, and response delivery. After each trial, the protected holding register (%MW0) was read back through an independent connection to verify state mutation.

Table 14 details performance across 1,000 trials for each scenario.

Baseline Modbus/TCP write latency without SENTINEL averaged 13.25 ms (P50: 13.46 ms, P95: 14.33 ms, P99: 14.40 ms). For valid writes, protected latency achieved P50 of 17.52 ms (P99: 18.73 ms). For each trial index i , paired overhead was calculated as $\Delta t_i = t_{\text{protected},i} - t_{\text{baseline},i}$; the reported 4.06 ms value is the median of the resulting 1,000 paired differences (P95: 4.23 ms). The 4.06 ms paired median end-to-end overhead is consistent in magnitude with the 5.10 ms median preprocessing-and-scoring microbenchmark, although the measurements cover different execution paths. Rejected and blocked scenarios short-circuited execution at the policy or gateway stage in under 2.51 ms P50 without initiating socket writes to the OpenPLC.

Scenario	Rejection stage	Register modified?	P50	P95	P99
Valid operator write	Complete SENTINEL path	Yes	17.52	18.56	V
Model B modifies value	Gateway HMAC check	No	2.51	2.74	2.76
Replayed capability	Gateway Nonce cache	No	2.51	2.75	2.77
Unauthorized role	Model A hard policy	No	2.50	2.75	2.77
Slow setpoint drift	Model A Scorer / Deception	No	2.50	2.74	2.77
Model B retargets register	Gateway target check	No	2.49	2.74	2.76

Table 14: MODBUS/TCP Testbed Scenario Results (1,000 Trials Per Scenario)

Table XV reports sequence-level evaluation across 100 adaptive slow-drift attack sessions. Because Model A aggregates rate of change across the 48-command window, drift sessions were

diverted into deception at a median of 18 commands. Across all 100 sequences, 0% of sub-threshold setpoint changes mutated the live OpenPLC register prior to containment.

Slow-drift evaluation metric	Observed result
Attack sequences evaluated	100
Median commands to deception diversion	18
P95 commands to deception diversion	21
Maximum cumulative live setpoint shift	0.00
Sequences reaching target physical shift	0 / 100
Unauthorized protected-register modifications	0

Table 15: Sequence-Level Slow Setpoint Drift Testbed Evaluation

8. Security Discussion

8.1. Compromised Model B

Model B processes untrusted network inputs, converts public protocols into canonical form, and maintains transport state. If an adversary achieves remote code execution in Model B, they can alter payloads, reorder commands, or drop requests. However, because Model B lacks the HMAC key, it cannot produce an HMAC-authenticated capability generated by Model A. An altered write payload causes a capability verification failure at the OT enforcement point.

8.2. Replay and Substitution

Binding the token to the command value, target, operation, worker, role, nonce, expiry, and policy version prevents payload substitution and cross-target reuse. A nonce cache prevents a valid token from being used twice during its 800 ms lifetime. The command and capability channel should additionally use mutually authenticated and confidential transport to reduce capability interception and first-use token-race attacks. Deployments must define bounded clock skew, key rotation, cache persistence after gateway restart, and safe behavior during policy-version changes.

8.3. Deception Risks

An attacker may fingerprint a decoy through timing, process physics inconsistencies, cross-session observations, or protocol details. SENTINEL therefore treats deception as containment and evidence collection, not as a guarantee that the adversary cannot identify the decoy. Higher-fidelity process simulation and network-stack equalization are necessary for advanced adversaries [16].

8.4. Adversarial Learning and Poisoning

The reference scorer is supervised and can be affected by distribution shift, label error, or poisoning. Evidence collected from a suspicious session is never promoted automatically into the trusted baseline. Model updates should use versioned, reviewed data; performance should be monitored by worker, device, and plant mode; and rollback should be possible. The present work evaluates a shifted test set but does not provide certified robustness.

8.5. Empirical Adaptive-evasion Analysis

We implemented an optimization agent that crafts subthreshold setpoint-drift sequences against the trained SEN-TINEL random-forest scorer. For target physical perturbations $\Delta V \in \{0.20, 0.40, 0.60, 0.80, 1.00\}$, the adaptive search agent attempts to maximize cumulative setpoint shift while keeping the maliciousness score strictly below the clear threshold $\tau_{\text{clear}} = 0.45$.

Under the evaluated synthetic generator and trained scorer, adaptive attacks required progressively more commands as the requested cumulative perturbation increased: achieving a modest perturbation of $\Delta V = 0.20$ required 18 sequential subthreshold commands, while larger setpoint shifts ($\Delta V \geq 0.60$) required at least 64 commands in the evaluated runs, as plotted in Fig. 4. When the attacker attempts to accelerate the setpoint perturbation rate, the windowed linear slope ∇v_j and cross-sensor residual $\hat{r}_i$ caused the evaluated sequences to cross τ_{clear} , resulting in diversion before the requested cumulative perturbation was completed. This imposed a substantial ratelimiting penalty on the evaluated adaptive attack instances.

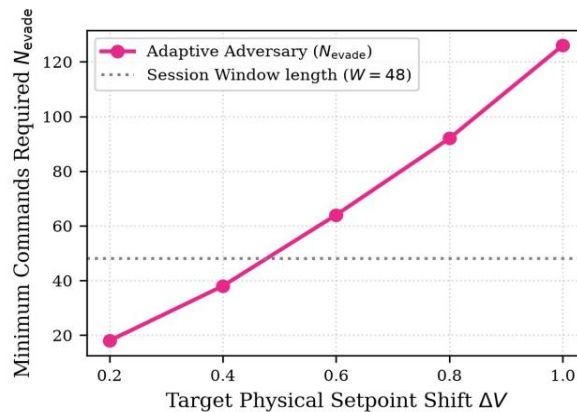


Figure 4: Empirical evasion penalty curve showing minimum commands N_{evade} required by an adaptive adversary to achieve setpoint shift ΔV without exceeding $\tau_{\text{clear}}=0.45$

8.6. Availability, Keys, and Protocol Integration

Mandatory mediation creates an availability dependency on Model A and the enforcement point. Flooding, session-state exhaustion, and repeated scoring should be limited through early validation, per-identity quotas, bounded state, priority classes, and replicated decision services. Remote writes fail closed, while local safety and manual-control paths remain independent.

HMAC protects integrity only while the shared secret remains confidential. Production deployments should use protected key storage, key identifiers, rotation, and restart-safe nonce handling; asymmetric signatures may be substituted when shared-key distribution is unsuitable. Protocol adapters must also produce an unambiguous canonical command. For example, a Modbus binding should cover unit identifier, function, address, and value, while an OPC UA binding should cover endpoint, node, operation, and typed value. Audit records should preserve the command hash, identity context, model and policy versions, score, action, and verifier outcome.

9. Threats to Validity and Limitations

The synthetic labels represent designed attack mechanisms, not ground-truth malicious intent in a real organization. Origin trust, process residual, role permission, and operating modes are simplified numeric signals. The generator, feature design, and model were developed within the same study, creating generator-model coupling. Complete-session splitting, benign transients, and a hardened test transformation reduce leakage, but they do not provide the independence of a separately collected plant dataset. Bootstrap intervals quantify only resampling uncertainty in the fixed 600-session test set.

The evaluation is command-centric and does not reproduce PLC scan cycles, network partitions, electromagnetic interference, operator workload, or protocol-specific behavior. The latency benchmark excludes network, queue, audit persistence, and device response time, and the verifier is modeled at a gateway rather than executed in PLC firmware. Adaptive, coordinated, and perfectly indistribution long-horizon attacks remain outside the demonstrated coverage.

10. Future Work and Validation Roadmap

The next stage is hardware-in-the-loop validation using a physical PLC or high-fidelity testbed with real protocol timing and process constraints. A complete study should measure false diversions and blocks per hour over long benign runs; end-to-end latency through protocol ingress, audit persistence, gateway verification, and PLC acknowledgement; concurrent replay races, clock skew, key rotation, restart behavior, and Model A failover. Model evaluation should add multiple seeds, unseen workers and devices, independently designed attacks, and process-aware graph residuals for insider drift. Deception should be tested for timing fidelity, cross-sensor consistency, multi-session isolation, and fingerprint resistance.

11. Conclusion

SENTINEL combines mandatory write mediation, deterministic policy checks, temporal trust scoring, uncertainty-aware deception, and target-bound cryptographic authorization. In the reproducible synthetic study, the scorer achieved 95.33% accuracy, 90.82% macro-F1, and a 0.28% benign FPR. The selected triage policy contained 98.33% of attack sessions while approving 99.17% of benign sessions, and all 14,000 evaluated token-misuse attempts were rejected. Insider drift remains difficult to attribute separately from slow evasion, although its maliciousness score is sufficient for containment.

The results support architectural feasibility under the stated simulation assumptions, not field readiness or safety certification. Real protocol traffic, independent attack generation, longduration benign operation, and hardware-in-the-loop testing remain necessary. The artifact package includes the generator, trained scorer, metrics, token and latency tests, and figure scripts.

References

1. Stouffer, K., Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., ... & Thompson, M. (2023). Guide to operational technology (ot) security.
2. Urbina, D. I., Giraldo, J. A., Cardenas, A. A., Tippenhauer, N. O., Valente, J., Faisal, M., ... & Sandberg, H. (2016, October). Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 1092-1105).
3. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. *Proc. NDSS*, 2018.
4. Deng, A., & Hooi, B. (2021, May). Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, No. 5, pp. 4027-4035).
5. Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2020, August). Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 3395-3404).
6. Tuli, S., Casale, G., & Jennings, N. R. (2022). Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*.
7. The HoneyNet Project, "Conpot ICS/SCADA HoneyPot Documentation," version 0.6.0. [Online].
8. Maesschalck, S., Giotsas, V., Green, B., & Race, N. (2022). Don't get stung, cover your ICS in honey: How do honeypots fit within industrial control system security. *Computers & Security*, 114, 102598.
9. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero trust architecture. *NIST special publication*, 800(207), 1-52.
10. Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection

-
- systems. *Ieee Access*, 8, 165130-165150.
11. Mathur, A. P., & Tippenhauer, N. O. (2016, April). SWaT: A water treatment testbed for research and training on ICS security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)* (pp. 31-36). IEEE.
 12. Goh, J., Adepu, S., Junejo, K. N., & Mathur, A. (2016, October). A dataset to support research in the design of secure water treatment systems. In *International conference on critical information infrastructures security* (pp. 88-99). Cham: Springer International Publishing.
 13. Williams, J. J., Edwards, M., & Gardiner, J. (2025, July). Time-to-lie: Identifying industrial control system honeypots using the internet control message protocol. In *2025 IEEE 45th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 806-812). IEEE.
 14. Anagnostakis, K. G., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., & Keromytis, A. D. (2005). Detecting targeted attacks using shadow honeypots.
 15. Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC: Keyed-hashing for message authentication* (No. rfc2104).
 16. Vassilev, A., Oprea, A., Fordyce, A., Anderson, H., Davies, X., & Hamin, M. (2025, March). *Adversarial machine learning: A taxonomy and terminology of attacks and mitigations* (Report No. NIST AI 100-2e2025). National Institute of Standards and Technology.

Copyright: ©2026 Mohammad Salman Khan, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.