

A Reproducible Zero Trust Testbed for Dynamic Trust Evaluation Using Emulation and Virtualization Tools

Tiberius Nkinyili,
0000-0002-1798-3176,
Strathmore University,
Nairobi, Kenya
Email: ttabulu@strathmore.edu

Vincent Omwenga,
0000-0002-9443-500X,
Strathmore University,
Nairobi, Kenya.
vomwenga@strathmore.edu,

Solomon Ogara,
0000-0002-3321-0448,
Jaramogi Oginga Odinga
University of Science and
Technology, Kisumu Kenya.
sogara@jooust.ac.ke.

Abstract—The practical evaluation of Zero Trust Architecture (ZTA) remains constrained by the lack of accessible, reproducible testbeds supporting dynamic trust experimentation. This paper presents a lightweight, containerized Zero Trust testbed integrating a Software Defined Perimeter (SDP), Software Defined Networking (SDN), network emulation, and policy-as-code enforcement. A pluggable Python trust engine implements progressively sophisticated trust models, culminating in Dempster–Shafer evidential fusion with temporal decay and residual inertia. Evaluation across six representative enterprise scenarios demonstrates low policy evaluation latency, modest session setup overhead, linear scalability, and rapid breach containment, while remaining deployable on commodity hardware. By combining reproducibility, open-source components, and advanced trust evaluation within a single platform, the proposed testbed provides an accessible environment for research, education, and the experimental validation of Zero Trust mechanisms.

Index Terms—Zero Trust testbed, Software-Defined Perimeter (SDP), Software-Defined Networking (SDN), Mininet, GNS3, Docker, dynamic trust evaluation, virtualization, emulation, containerization, reproducibility.

I. INTRODUCTION

ZERO Trust Architecture (ZTA) has become the dominant security paradigm for modern enterprise networks due to the erosion of traditional network perimeters, widespread adoption of cloud services, and the proliferation of mobile and IoT devices [1], [2]. Guided by the principle of “never trust, always verify,” ZTA enforces continuous authentication, authorization, and monitoring of every access request [3]. Frameworks such as [4], [5] and [6], and recent cybersecurity initiatives have accelerated its adoption. However, despite the maturation of ZTA frameworks [6], a significant gap remains between theoretical trust models and their practical validation. This gap spans three dimensions: algorithmic opacity, where trust models remain mathematically specified but not reproducible [7]; infrastructure inaccessibility, due to the complexity of deploying realistic ZTA environments [8]; and scenario non-standardization, where poorly specified evaluation conditions hinder reproducibility and cross-study comparison [7], [9].

This paper addresses these challenges through a reproducible, lightweight testbed that maps the NIST architecture onto open-source, containerized components deployable on a commodity workstation. The testbed provides executable

trust engines, Infrastructure-as-Code deployment, and standardized scenarios with published telemetry and empirical datasets. Consistent with the resource-constrained philosophy of prior work [1], it enables reproducible experimentation with dynamic trust models using only commodity hardware.

A. Contributions

The main contributions of this paper are: (1) a modular, lightweight Zero Trust testbed integrating Software-Defined Perimeter, Software-Defined Networking, and containerized services using open-source components; (2) a pluggable Python-based trust engine supporting progressive policy models, including static weighting and Dempster–Shafer evidence fusion with temporal decay and residual inertia; (3) experimental validation across six enterprise scenarios evaluating latency, breach containment, resource utilization, false-positive rates, and scalability; and (4) a reproducible infrastructure-as-code implementation to support replication and future research.

The remainder of the paper reviews related work (II), presents the system architecture (III) and implementation (IV), describes the experimental methodology (V), results (VI), and discussions (VII), and concludes with limitations (VIII), reproducibility (IX) and future research directions (X).

II. RELATED WORK

A. Zero Trust Testbeds and Dynamic Trust Evaluation

Literature contains several efforts to implement Zero Trust principles in testbed environments. Reference [1] demonstrated combining Software Defined Perimeter (SDP) with Software Defined Networking (SDN) and lightweight virtualization for zero trust experimentation with Mininet as a key enabler for resource-constrained prototyping. Reference [10] extended this foundation by building an SDP for network introspection, demonstrating how SDP components map to ZTA concepts in [3]. Their work validated that SDP 2.0 specifications directly support ZTA’s Single Packet Authorization (SPA). Reference [11] investigated identity-aware SDN controllers, while various commercial implementations have demonstrated Zero Trust Network

Access (ZTNA) as alternatives to traditional VPNs [12]. However, these efforts share common limitations: a lack of dynamic trust evaluation, dependence on proprietary components, or failure to provide reproducible artifacts.

Within the CARTA paradigm [13], AI and machine learning support continuous trust assessment through behavioral modelling, adaptive trust scoring, and predictive threat detection [14]. However, these approaches typically depend on labelled training data and often lack interpretability. The proposed engine uses deterministic variance-based weighting for unstable evidence, providing transparent and verifiable trust decisions suitable for adversarial environments. While explainability is prioritized, optimization of the variance parameter through reinforcement learning is identified as future work [15], [16].

B. Simulation, Emulation, and Virtualization Approaches

The choice between simulation, emulation, and full virtualization has implications on testbed fidelity and resource requirements. Simulators model network behavior mathematically, offering high scalability but limited realism for protocol-specific and timing-sensitive behaviors. Emulators execute real network stacks in virtualized environments, providing a balance between fidelity and resource efficiency. Full virtualization runs unmodified operating systems and applications but incurs higher overhead. Table 1 compares common approaches with their respective strengths and limitations.

Table 1. Comparison of testbed approaches for zero trust experimentation

Approach	Examples	Fidelity	Resource Cost	Isolation	Reproducibility
Simulation	ns-3, OMNeT++	Low (models)	Very low	Complete	High
Lightweight emulation	Mininet	Medium (Linux netns)	Low	Partial	High
Full emulation	GNS3, Eve-NG	High (real images)	Medium-High	Good	Medium
Type 2 virtualization	Virtual Box, QEMU/KVM	Very high	High	Strong	Medium
Type 1 virtualization	VMware ESXi, Xen	Native	Very high	Strong	Low

Mininet remains the de facto standard for SDN prototyping and emulation of large-scale networks through lightweight Linux network namespaces. It supports efficient experimentation in resource-constrained environments while achieving aggregated bandwidths exceeding 10 Gb/s with over 1,000 hosts on commodity hardware [17]. Although its lightweight virtualization offers weaker isolation and introduces security challenges compared to Type 1 hypervisors [18], these limitations are generally acceptable for zero trust experimental

tion. Applications requiring hardware-level isolation or proprietary device emulation may instead benefit from platforms such as GNS3 or full virtualization solutions.

C. Software Stack for Zero Trust Experimentation

Numerous closed and open-source projects have been used to implement parts of ZTA. Table 2 catalogs tools relevant to zero trust testbed construction, extending prior surveys with additional components for dynamic trust evaluation.

Table 2. Open-source software components for zero trust testbeds.

Component	Purpose
Keycloak	Identity Provider (OAuth2, OIDC, SAML)
Open Policy Agent (OPA)	Policy Decision Point (PDP)
Envoy Proxy	L7 Policy Enforcement Point (PEP)
OpenDaylight	SDN Controller (OpenFlow)
Open vSwitch (OVS)	OpenFlow-enabled virtual switch
Mininet	Network emulation
GNS3	Full network emulation
Docker	Container platform for applications
Pritunl Zero	Overlay zero trust server
OpenZiti	Zero trust networking overlay
SPIFFE/SPIRE	Identity providers for non-user components such as services and nodes
eBPF Cilium/Tetragon	Kernel level filters for processes and system calls
Kubernetes	Network orchestration

D. Trust Evaluation Models

Trust evaluation has evolved from static, to reputation-based, and dynamic models. Static approaches assign fixed weights to trust attributes regardless of signal quality, making them insensitive to changing operating conditions [9]. Reputation-based models leverage historical interactions to estimate trust [19], but do not effectively distinguish transient compromise from persistent environmental variability. Dynamic approaches employ adaptive weighting, including entropy-based methods and machine-learning [15], [20].

E. Dempster-Shafer Theory in Network Security

Dempster-Shafer theory [21] has been applied to intrusion detection [22], anomaly detection [23], and trust management [24]. Unlike Bayesian methods, it explicitly models epistemic uncertainty through belief mass assigned to a frame of discernment, avoiding forced probability assignments with incomplete evidence. However, existing approaches assume uniform source reliability, neglect the relationship between signal variance and reliability [25], and fuse evidence within a single domain rather than across heterogeneous domains.

F. Temporal Trust Dynamics

Temporal trust employs linear or exponential decay to reduce the influence of stale evidence [26], [27]. While effective in capturing evidence freshness, they do not account for consistency of observations. Continuous Adaptive Risk

and Trust Assessment (CARTA) frameworks combine temporal and contextual information to balance trust revocation against prolonged trust retention [28]. The proposed ensemble model addresses this through a dual-horizon architecture that considers historical stability and recent observations.

G. Gap Analysis

No prior testbed integrates a trust engine supporting Dempster-Shafer fusion, variance-based weighting and temporal decay, with SDP/SDN enforcement in a reproducible, low-resource manner. Existing implementations either treat trust as static binary rules or fixed weights, depend on proprietary policy engines, or lack end-to-end integration across control and data planes. This paper directly addresses this gap.

III. TESTBED ARCHITECTURE AND DESIGN PRINCIPLES

The proposed testbed integrates Mininet for SDN-based network emulation, GNS3 for high-fidelity topology emulation and telemetry generation, and Docker for lightweight service virtualization. This hybrid architecture balances functional realism with the resource constraints of commodity hardware.

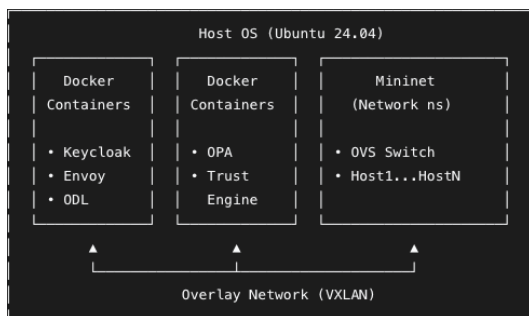
A. Design Principles

The architecture is guided by five principles: (1) operation on commodity hardware without proprietary software for resource-constrained philosophy; (2) decoupled Policy Decision Point (PDP) and Policy Enforcement Point (PEP) for modular trust-policy evaluation; (3) automated deployment through Docker Compose and infrastructure-as-code for reproducibility; (4) progressively increasing sophistication of trust models, from static policies to Dempster-Shafer evidence fusion with temporal decay; and (5) complete version-controlled reproducibility of the testbed.

B. Logical ZTA Component Mapping

Figure 1 presents the conceptual architecture, while Table 3 maps the implemented components to the logical entities defined in [3], demonstrating alignment with the Zero Trust

Figure 1. High-level architecture diagram (conceptual)



C. Physical and Virtualization Layers

The testbed is hosted on Ubuntu 24.04 LTS and is designed to operate on commodity hardware (minimum 4 CPU cores,

16 GB RAM, and 256 GB SSD). Mininet with Open vSwitch provides SDN-based network emulation, with optional GNS3 integration for scenarios requiring greater network fidelity. Docker and Docker Compose orchestrate the control-plane services including: Keycloak, Open Policy Agent (OPA), Envoy, and OpenDaylight thus providing lightweight isolation, portability, and reproducible deployment.

Table 3. Mapping of NIST ZTA components to testbed implementations

ZTA Component	Testbed Implementation	Role
Policy Decision Point (PDP)	Open Policy Agent (OPA) + Python trust engine	Computes trust scores and makes access decisions based on telemetry and policies.
Policy Administrator (PA)	OpenDaylight SDN controller + custom orchestration	Translates PDP decisions into network flow rules (OpenFlow) and service-mesh configurations.
Policy Enforcement Point (PEP)	Open vSwitch (L3), Envoy Proxy (L7), eBPF(cilium and tetragon)	Enforces allow/deny decisions at network and application layers.
Identity Provider (IdP)	Keycloak, SPIFFE/SPIRE	Provides authentication and device posture attributes (OAuth2/OIDC).
Telemetry Sources	Mininet hosts, OVS, Envoy, GNS3 nodes, Suricata, Zeek	Stream real-time metadata (identity, device health, network variance, application behaviour, NIDS data).
Subject (User/Device)	Mininet hosts or GNS3 virtual machines	Source of telemetry data
Resource	Containerised services (e.g., web servers, databases)	Source of telemetry data

IV. IMPLEMENTATION DETAILS

A. Testbed Preparation

The steps in Table 4 prepare a minimal installation for testbed deployment. These are later automated based on a yaml script for easier reproducibility.

Table 4. Testbed Deployment Workflow

Host Preparation

```

sudo apt update && sudo apt install -y git curl bridge-
utils net-tools

```

Resource utilization**vmstat-a****Docker**

```
sudo apt install -y docker.io docker-compose
sudo systemctl enable docker
sudo usermod -aG docker $USER
newgrp docker
```

OpenVswitch-PEP-Network Layer

```
sudo apt install -y openvswitch-switch
sudo systemctl start openvswitch-switch
ovs-vsctl show
```

Mininet- Emulator

```
sudo apt install -y mininet
sudo mn --test pingall
```

OpenDaylight

```
docker run -d \
    --name opendaylight \
    -p 8181:8181 -p 6653:6653 \
    opendaylight/odl:latest
```

Keycloak-Identity and Device Trust Anchor

```
docker run -d \
    --name keycloak \
    -p 8080:8080 \
    -e KEYCLOAK_ADMIN=admin \
    -e KEYCLOAK_ADMIN_PASSWORD=admin \
    quay.io/keycloak/keycloak:latest start-dev
```

B. Architectural Workflow

The proposed testbed follows a five-stage workflow. Firstly, Keycloak authenticates users, issuing JWTs containing identity and session claims. Secondly, the trust engine collects multi-domain telemetry (network traffic, device posture, application behaviour, and resource metadata). Thirdly, the Engine computes a trust score using the ensemble model. Fourthly, OPA evaluates the trust score against rego policy rego rules as in Table 5, to determine the access tier. Finally, the Policy Administrator enforces decisions through OpenDaylight and Open vSwitch's micro-segmentation and application-level access control.

The network is emulated in Mininet using a three-tier topology comprising corporate, DMZ, and remote segments interconnected through an OVS core switch. Fine-grained micro-segmentation is achieved through dynamically installed OpenFlow rules updated in response to trust evaluations. For reproducibility, Table 6 shows repository replication process.

Table 5. Access Tiers Rego policy

```
package trust.policy

default allow = false
default access_level = "no_access"
access_level = "full_access" {
    input.trust_score > 0.75
}
access_level = "limited_access" {
    input.trust_score >= 0.45
    input.trust_score <= 0.75
}
```

```
}
access_level = "no_access" {
    input.trust_score < 0.45
}
```

Table 6. Automated Deployment

```
# Clone repository and deploy
git clone https://github.com/[redacted]/dcta.git
cd dcta
docker compose up -d # IdP, OPA, ODL containers
sudo python3 deploy_mininet.py # Network topology
python3 run_ensemble_scenarios.py # Execute evaluation
```

C. Design Considerations and Future Extensions

The testbed employs Docker Compose to maximize reproducibility and minimize resource overhead, providing deterministic deployment and seamless integration with Mininet and OpenvSwitch. Although the architecture is readily portable to Kubernetes for production-scale deployments, Docker offers greater experimental repeatability.

V. VALIDATION SCENARIOS AND EXPERIMENTAL METHODOLOGY**A. Ensemble Trust Engine**

The proposed Trust Engine integrates variance-based dynamic weighting, Dempster-Shafer (DS) evidential fusion, and dual-horizon temporal trust dynamics to support adaptive trust evaluation in a Zero Trust environment. This paper focuses on their integration within a reproducible testbed. The ensemble trust score is computed as:

$$T_{\text{ensemble}}(t) = \underbrace{W_{\text{short}}(t) \cdot T_{\text{instant}}(t)}_{\text{Fresh Signal (Verification)}} + \underbrace{(1 - W_{\text{short}}(t)) \cdot T_{\text{prev}} \cdot D_{\text{long}}(\Delta t)}_{\text{Historical Inertia (Continuity)}}$$

Where T_{instant} represents the current trust obtained through DS fusion, W_{short} controls the influence of recent evidence, T_{prev} is the previous trust state, and D_{long} models long-term trust decay.

Trust is evaluated across four domains. For each domain, a sliding-window variance estimates signal reliability, with higher variance reducing the corresponding evidence weight before fusion. Domain trust scores are converted into DS Basic Probability Assignments (BPAs), enabling explicit representation of uncertainty and robust multi-domain evidence fusion. The resulting belief is transformed into an instantaneous trust score using the Pignistic probability. To support continuous authentication, the engine combines this instantaneous assessment with historical trust through dual-horizon temporal weighting, balancing responsiveness to recent events against resilience to transient fluctuations. The final ensemble trust score is mapped to configurable policy thresholds that grant Full, Limited, or No Access, enabling risk-adaptive access control under uncertainty.

B. Experimental Design

Five trust evaluation models of increasing sophistication were compared: Baseline (no policy enforcement), Single Do-

main (binary trust based on one attribute), Multi-Domain Hierarchical (static rule-based trust), Static Weighted (fixed trust weights), and the proposed Ensemble, which integrates variance-based adaptive weighting, Dempster-Shafer evidence fusion, temporal decay, and residual trust. They were evaluated using six representative enterprise network scenarios in Table 7: Corporate Office, Remote VPN, Public Wi-Fi, BYOD, Compromised Host, and Untrusted Device/Geofence. These scenarios span the full trust spectrum, varying domain trust levels and variance characteristics to represent stable, noisy, and adversarial environments. Each scenario was executed over 30 evaluation epochs (1-minute intervals), with domain scores generated through controlled randomization around predefined means to ensure reproducible variance profiles.

Table 7. Scenario Configuration Matrix

Scenario	Network Profile	Device Profile	Data/Identity	Application	Variance Characteristics	Ground Truth
Corporate Office	High (≈ 0.94)	High (≈ 0.95)	High (≈ 0.88)	High (≈ 0.90)	Stable ($\sigma^2 < 0.02$)	Full Access
Remote VPN	Mode rate (≈ 0.84)	High (≈ 0.98)	High (≈ 0.92)	High (≈ 0.91)	Variable ($\sigma^2 \approx 0.04$)	Full Access
Public Wi-Fi	Low (≈ 0.30)	Mode rate (≈ 0.75)	Mode rate (≈ 0.60)	Mode rate (≈ 0.72)	Chaotic ($\sigma^2 > 0.20$)	Limited Access
BYOD	High (≈ 0.89)	Low (≈ 0.40)	Mode rate (≈ 0.49)	Mode rate (≈ 0.62)	$\sigma^2 \approx 0.15$	Limited Access
Compromised Host	Low (≈ 0.25)	Low (≈ 0.21)	Low (≈ 0.21)	Low (≈ 0.27)	Chaotic across all domains	No Access
Untrusted Device/Geofence	Mode rate (≈ 0.31)	Low (≈ 0.34)	Mode rate (≈ 0.30)	Low (≈ 0.30)	Unstable ($\sigma^2 \approx 0.10$)	No Access

C. Evaluation Methodology

Performance was assessed using classification accuracy, convergence time, ensemble stability, per-evaluation latency, and breach containment time. Each scenario was repeated across 50 independent simulation runs with different random

seeds, and results are reported as mean $\pm$ standard deviation. Statistical significance was evaluated using the Wilcoxon signed-rank test ($p < 0.01$), with effect sizes measured using Cliff's delta. The ensemble model employed a variance penalty factor of $\alpha = 10.0$ and short- and long-term temporal decay rates of $\lambda = 3.0$ and $\lambda = 0.5$, respectively.

D. Implementation

The prototype was implemented in Python 3.12 using modular components:

- `dynamic_trust_weighting.py`: Variance computation, dynamic weight calculation, intra-domain score aggregation.
- `weighted_belief_fusion.py`: DS mass function construction, Dempster combination rule, Pignistic transformation.
- `ensemble_trust_simulator.py`: Dual-horizon temporal integration, ensemble computation.
- `ds_utils.py`: Generic Dempster-Shafer library (mass function manipulation, belief calculation).
- `run_ensemble_scenarios.py`: Scenario orchestration, data logging, and visualisation.

VI. EXPERIMENTAL RESULTS

A. Trust Classification Performance

The proposed Ensemble Trust Engine correctly classified all six evaluation scenarios at steady state ($t \geq 10$) over 50 independent runs as in Table 8. High-trust environments (Corporate Office and Remote VPN) consistently received Full Access, uncertain environments (Public Wi-Fi and BYOD) were assigned Limited Access, while compromised and untrusted scenarios were correctly denied access. Trust trajectories remained stable throughout evaluation without crossing decision thresholds, demonstrating robust and consistent policy enforcement.

The results further show that variance-based adaptive weighting effectively suppresses unreliable domains while allowing stable evidence to dominate trust estimation. Consequently, transient network instability (Remote VPN and Public Wi-Fi) did not result in unnecessary access revocation, whereas persistent multi-domain compromise consistently produced sustained denial. The ensemble temporal model progressively reinforced trust under stable conditions while preventing rapid oscillations caused by short-term fluctuations.

Table 8. Ensemble Trust Scores and Access Decisions

Scenario	Tensem ble ($t=0$)	Tensem ble ($t=15$)	Tensem ble ($t=29$)	Decision
Corporate Office	0.795 $\pm$ 0.003	0.796 $\pm$ 0.002	0.792 $\pm$ 0.002	Full Access $\checkmark$
Remote VPN	0.787 $\pm$ 0.005	0.787 $\pm$ 0.004	0.783 $\pm$ 0.004	Full Access $\checkmark$

Public Wi-Fi	0.570±0.012	0.606±0.011	0.604±0.010	Limited Access ✓
BYOD	0.558±0.015	0.607±0.013	0.599±0.012	Limited Access ✓
Compromised Host	0.227±0.009	0.302±0.008	0.299±0.007	No Access ✓
Untrusted Device/Geofence	0.360±0.011	0.343±0.010	0.346±0.009	No Access ✓

B. Performance Evaluation

The complete trust evaluation pipeline required an average of 18.5 ms per evaluation cycle, satisfying the design target of <20 ms as in Table 9. Mathematical trust computation accounted for 6.9 ms, with memory state management representing the primary overhead. Total session establishment latency increased from 8–12 ms (baseline) to 23–32 ms with the Ensemble model, remaining well within acceptable enterprise response times.

During simulated compromise scenarios, trust values fell below the revocation threshold within one evaluation interval, enabling access revocation in less than 15 seconds, compared with several minutes for baseline and static trust approaches. This substantially reduces the attack window for lateral movement while maintaining responsiveness.

C. Stability and Scalability

Compared with a spatial-only Dempster-Shafer model, the Ensemble approach reduced trust score volatility by 33–56% across all scenarios ($p < 0.001$, Cliff's $\delta > 0.80$), demonstrating statistically significant improvements in access stability and reducing false-positive revocations caused by transient fluctuations.

The prototype remained lightweight, consuming approximately 3.2 GB RAM and 10–15% CPU at idle, increasing to 4.1 GB RAM and 35–40% CPU under 100 concurrent sessions. Trust reevaluation scaled linearly (≈ 0.2 ms per additional session), while SDN flow installation remained below 50 ms, confirming suitability for small- to medium-scale enterprise Zero Trust deployments.

Table 9. Latency Breakdown per Evaluation Epoch

Component	Complexity	Latency (ms)
Variance computation	$O(N)$ per domain, $N=10$	2.1
Weight normalization	$O(D)$, $D=4$	0.3
Mass function construction	$O(D)$, $D=4$	0.4
DS pairwise fusion (3 iterations)	Closed form for binary frame	3.8

Pignistic transformation	$O(1)$	0.1
Temporal integration (ensemble mixture)	$O(1)$	0.2
memory state read/write	Network I/O	8.4
OPA policy evaluation	Rego evaluation	3.2
Total		18.5

VII. DISCUSSION

The proposed testbed demonstrates that integrating identity-aware SDP enforcement with SDN enables continuous, low-latency micro-segmentation driven by dynamic trust evaluation. The results further demonstrate that trust should be modelled as a dynamic continuous property reflecting both historical stability and current behaviour.

Compared with existing Zero Trust testbeds, the proposed platform uniquely provides an integrated dynamic trust engine, multiple policy models, reproducible Infrastructure-as-Code deployment, and comprehensive empirical evaluation, including latency, false-positive, and breach-containment analyses as in Table 10. The substantially reduced false positives illustrate an effective security–usability trade-off. These capabilities establish a practical and extensible research platform for evaluating adaptive trust mechanisms under standardized conditions.

Table 10. Comparison of Results with Other works

Feature	Horne [1]	Lefebvre [10]	This work
SDP/SDN integration	✓	✓	✓
Dynamic trust engine	✗	✗	✓ (DS + variance + decay)
Multiple policy models	✗	Partial	✓ (6 models)
Reproducible artifacts	Partial	✗	✓ (Full IaC)
Empirical latency measurements	✗	✓	✓
False positive analysis	✗	✗	✓
Breach containment measurement	✗	✗	✓

VIII. LIMITATIONS AND DESIGN IMPROVEMENTS

The proposed framework is modular and vendor-agnostic, enabling interoperability with different implementations of identity providers, SDN controllers, and policy enforcement mechanisms without modifying the trust engine. Evaluation was conducted in a containerized Mininet environment using simulated telemetry and binary trust states, limiting direct generalization to production-scale deployments. Although the

framework demonstrates scalability for small and medium-sized enterprise environments, further validation using real security telemetry, hardware trust anchors (e.g., TPM 2.0), multi-state trust models, and distributed deployments is required. Design improvements to address these include integrating SPIFFE/SPIRE for workload identity, eBPF-based enforcement (e.g., Cilium and Tetragon), and real network telemetry from Zeek and Suricata to evaluate trust using operational protocol metadata, behavioral anomalies, and intrusion alerts. These are summarized in *Table 11*.

Each technology addresses a specific limitation identified in the paper while preserving the testbed's design principles: open-source, reproducible, and lightweight.

Table 11. Future Improvements to Challenges and Limitations

Technology	Testbed Layer	Current State	Extension
NIST SP 1800-35 [6]	Standards	SP 800-207 mapping	Cross-reference with 1800-35 builds
AI/ML scoring [20], [29], [30]	Trust Engine	Deterministic variance weighting	Adaptive α via RL
Docker [31], [32]	Orchestration	Docker Compose	K8s for production scale
SPIFFE/SPIRE [33], [34]	Identity	Keycloak (human IdP)	Machine identity + mTLS
eBPF/Cilium [35], [36]	Enforcement	OVS + Envoy	Kernel-level, sub-ms
Zeek + Suricata [37], [38]	Telemetry	Simulated Gaussian	Real network metadata + IDS

IX. REPRODUCIBILITY AND REUSABILITY

A. Code Repository Structure

To facilitate reproducibility, all source code, configuration files, topology definitions, policy scripts, and simulation outputs are provided in a public repository [39].

The repository includes implementations of the baseline, temporal decay, and ensemble trust models, together with automation scripts for provisioning the Docker–Mininet testbed and reproducing the experimental scenarios.

The testbed is designed to be easily extended: researchers can modify Rego policies, change trust engine parameters (decay rates, weights), add new telemetry sources, or integrate additional enforcement points (e.g., a service mesh).

X. CONCLUSION AND FUTURE WORK

This paper presented a lightweight, reproducible Zero Trust testbed that bridges theoretical trust evaluation with

operational enforcement through an open-source, containerized implementation aligned with NIST SP 800-207. The architecture integrates SDP-based identity verification, SDN-enabled microsegmentation, and an Ensemble Trust Engine combining variance-based adaptive weighting, Dempster-Shafer evidential fusion, and dual-horizon temporal dynamics to support continuous, context-aware access control.

Experimental evaluation demonstrated that the proposed framework correctly enforced dynamic access policies while maintaining low computational overhead (approximately 18.5 ms per evaluation cycle) and significantly reducing trust-score volatility and breach containment time compared with static approaches. The modular separation of the trust engine from policy enforcement proved particularly effective, enabling independent evolution of trust algorithms without reconfiguring the control plane, while the containerized design ensures reproducibility on commodity hardware.

Future work will extend the framework with hardware-backed device attestation [40], reinforcement learning for adaptive trust parameter tuning, multi-state trust models, cloud-scale deployments [30], [41], and formal policy verification to further improve scalability, robustness, and real-world applicability.

XI. ACKNOWLEDGEMENT

During the preparation of this work, the authors used Deepseek to summarize and aggregate individual docker container configurations to one yaml script for all containers and Claude (opus 4.6) to check inter-chapter cohesion and narrative flow. The authors take full responsibility for the content of this publication.

REFERENCES

- [1] D. Horne, “Leveraging Software Defined Perimeter (SDP), Software Defined Networking (SDN), and Virtualization to Build a Zero Trust Testbed with Limited Resources,” 2022.
- [2] D. T. Van, T. B. Hung, T. D. Thang, and N. K. Giao, “Application of Machine Learning in Malicious IoT Classification and Detection on Fog-IoT Architecture,” presented at the The Seventh International Conference on Research in Intelligent and Computing in Engineering, Feb. 2022, pp. 299–303. doi: 10.15439/2022R39.
- [3] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “NIST Special Publication 800-207 -Zero Trust Architecture.” Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, Aug. 10, 2020. doi: <https://doi.org/10.6028/NIST.SP.800-207>.
- [4] R. Ward and B. Beyer, “BeyondCorp: A New Approach to Enterprise Security,” *login.*, vol. Vol. 39, No. 6, pp. 6–11, 2014.
- [5] “Executive Order 14028,” Improving the Nation’s Cybersecurity. Accessed: Jun. 29, 2026. [Online]. Available: <https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity>
- [6] National Institute of Standards and Technology, “Implementing a Zero Trust Architecture,” U.S. Department of Commerce, NIST Special Publication 1800-35, 2025.
- [7] N. Ngema, B. Nleya, and R. C. Maswanganyi, “A Review of Zero Trust Architecture: Principles, Applications, and Implementation Challenges in Communication, Navigation, and Surveillance (CNS) Systems,” *Sensors*, vol. 26, no. 12, 2026, doi: 10.3390/s26123813.
- [8] R. Vaughn, T. Adewale, and J. Ajayi, “Zero-Trust Architectures for Securing Cloud-Native Infrastructure,” Nov. 2024.

- [9] N. F. Syed, S. W. Shah, A. Shaghghi, A. Anwar, Z. Baig, and R. Doss, "Zero Trust Architecture (ZTA): A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 57143–57179, 2022, doi: 10.1109/ACCESS.2022.3174679.
- [10] M. Lefebvre, S. Nair, D. W. Engels, and D. Horne, "Building a Software Defined Perimeter (SDP) for Network Introspection," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2021, pp. 91–95. doi: 10.1109/NFV-SDN53031.2021.9665152.
- [11] Y. Palmo, S. Tanimoto, H. Sato, and A. Kanai, "A Consideration of Scalability for Software Defined Perimeter Based on the Zero-trust Model," in *2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2021, pp. 717–724. doi: 10.1109/IIAI-AAI53430.2021.00127.
- [12] J. Adorno, "Evolution of Secure Access: From VPN to SDP-Enabled Zero Trust Network Access (ZTNA) | Zscaler." Accessed: May 19, 2026. [Online]. Available: <https://www.zscaler.com/blogs/product-insights/evolution-secure-access-vpn-sdp-enabled-zero-trust-network-access-ztna>
- [13] Trio.so, "CARTA: An Overview of Gartner's Continuous Adaptive Risk and Trust Assessment," Trio Security, 2025.
- [14] A. Tripathi *et al.*, "Real Time Adaptive Access Control with Behavioral Analytics for Enhanced Cybersecurity in IoT and Cloud Systems," presented at the The Ninth International Conference on Research in Intelligent Computing in Engineering, May 2025, pp. 151–155. doi: 10.15439/2024R75.
- [15] S. Munasinghe, N. Piyaarathna, E. Wijerathne, U. Jayasinghe, and S. Namal, "Machine Learning Based Zero Trust Architecture for Secure Networking," in *2023 IEEE 17th International Conference on Industrial and Information Systems (ICIIS)*, 2023, pp. 1–6. doi: 10.1109/ICIIS58898.2023.10253610.
- [16] M. Aaqib, A. Ali, L. Chen, and O. Nibouche, "Behaviour-based trust assessment for the Internet of Things systems using multi-classifier ensemble learning and Dempster-Shafer fusion," *Neural Computing and Applications*, vol. 37, no. 26, pp. 22191–22214, Sep. 2025, doi: 10.1007/s00521-025-11273-8.
- [17] B. Lantz and B. O'Connor, "A Mininet-based Virtual Testbed for Distributed SDN Development," *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 365–366, Aug. 2015, doi: 10.1145/2829988.2790030.
- [18] O. Flauzac, E. Robledo, and F. Nolot, "Is Mininet the Right Solution for an SDN Testbed?," Dec. 2019, pp. 1–6. doi: 10.1109/GLOBE-COM38437.2019.9013145.
- [19] F. Ullah *et al.*, "Deep Trust: A Novel Framework for Dynamic Trust and Reputation Management in the Internet of Things (IoT) Based Networks," *IEEE Access*, vol. PP, pp. 1–1, Jan. 2024, doi: 10.1109/ACCESS.2024.3409273.
- [20] D. Haider, S. Mushtaq, H. Ali, and M. Su'ud, "Enhancing Zero Trust Cybersecurity using Machine Learning and Deep Learning Approaches," *Journal of Informatics and Web Engineering*, vol. 4, pp. 24–34, Oct. 2025, doi: 10.33093/jiwe.2025.4.3.2.
- [21] G. Shafer, "A Mathematical theory of Evidence," 1976.
- [22] M. Maliha and M. Atiquzzaman, "Q-ID: A Reinforcement Learning Framework for Adaptive Intrusion Detection," presented at the 20th Conference on Computer Science and Intelligence Systems (FedC-SIS), Kraków, Poland, Oct. 2025, pp. 35–42. doi: 10.15439/2025F1820.
- [23] P. D. J. Fidalgo, A. Pasic, and S. G. Zarzosa, "Dynamic Threat Intelligence for Improvement of Resilience of Critical Infrastructure During Pandemics," presented at the 19th Conference on Computer Science and Intelligence Systems (FedCSIS), Belgrade, Serbia, Oct. 2024, pp. 591–596. doi: 10.15439/2024F8106.
- [24] N. Wang and D. Wei, "An Adaptive Dempster-Shafer Theory of Evidence Based Trust Model in Multiagent Systems," *Applied Sciences*, vol. 12, no. 15, p. 7633, Jul. 2022, doi: 10.3390/app12157633.
- [25] L. Xiong, X. Su, and H. Qian, "Conflicting evidence combination from the perspective of networks," *Inf. Sci.*, vol. 580, no. C, pp. 408–418, Nov. 2021, doi: 10.1016/j.ins.2021.08.088.
- [26] H. Wang, J. Jiang, and W. Li, "A Dynamic Trust Model Based on Time Decay Factor," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)*, 2018, pp. 2048–2051. doi: 10.1109/SmartWorld.2018.00343.
- [27] J. D. L. C. Kiguigouléli Ki and B. Zerbo, "Modeling Dynamic Trust for Cooperative Operations in Edge Computing Environments," in *2025 IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD)*, 2025, pp. 1–8. doi: 10.1109/MNE3SD67637.2025.11323241.
- [28] IBM Security, "Cost of a Data Breach Report 2024," p. 46, 2024.
- [29] M. Ahmed, M. Mark, and M. Joy, "Intelligent Trust: Leveraging AI for Dynamic Policy Orchestration in Zero Trust Security Architectures," *International Journal of Artificial Intelligence*, Aug. 2021.
- [30] V. Kalekar, "Uncertainty-Aware Reinforcement Learning for Zero Trust: An Empirical Evaluation of PPO, MC Dropout, and Bayesian Neural Networks," in *Proceedings of the 12th International Conference on Information Systems Security and Privacy*, Marbella, Spain: SCITEPRESS - Science and Technology Publications, 2026, pp. 505–514. doi: 10.5220/0014245800004061.
- [31] Dockerdocs, "DockerDocs Manuals," Docker Documentation. Accessed: Apr. 14, 2026. [Online]. Available: <https://docs.docker.com/desktop/>
- [32] D. Silva, J. Rafael, and A. Fonte, "Toward Optimal Virtualization: An Updated Comparative Analysis of Docker and LXD Container Technologies," *Computers*, vol. 13, no. 4, p. 94, Apr. 2024, doi: 10.3390/computers13040094.
- [33] "The SPIFFE Standard | SPIFFE." Accessed: Jun. 29, 2026. [Online]. Available: <https://spiffe.io/docs/latest/spiffe-specs/>
- [34] L. C. Cardoso, M. A. Marques, P. H. B. Correia, H. Z. Cochak, C. C. Miers, and M. A. Simplicio, "Next-Generation SPIFFE/SPIRE Identity Management Systems with Post-Quantum Cryptography Algorithms," in *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2025, pp. 154–163. doi: 10.1109/CCGRID64434.2025.00033.
- [35] "eBPF Security: Top 5 Use Cases, Challenges & Best Practices." Accessed: Jun. 29, 2026. [Online]. Available: <https://www.oligo.security/academy/ebpf-security-top-5-use-cases-challenges-and-best-practices>
- [36] "Installation and Configuration," Tetragon - eBPF-based Security Observability and Runtime Enforcement. Accessed: Jun. 29, 2026. [Online]. Available: <https://tetragon.io/docs/installation/>
- [37] "Zeek: Get Zeek," Zeek. Accessed: Jun. 29, 2026. [Online]. Available: <https://zeek.org/get-zeek/>
- [38] "2. Quickstart guide — Suricata 9.0.0-dev documentation." Accessed: Jun. 29, 2026. [Online]. Available: <https://docs.suricata.io/en/latest/quickstart.html>
- [39] "tnkinyili/DCTA: Dynamic and Context Aware Trust Algorithm for Zero Trust Enforcement in Heterogeneous Enterprise Environments," GitHub. Accessed: Apr. 16, 2026. [Online]. Available: <https://github.com/tnkinyili/DCTA>
- [40] M. Alawneh and I. M. Abbadi, "Integrating Trusted Computing Mechanisms with Trust Models to Achieve Zero Trust Principles," in *2022 9th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Milan, Italy: IEEE, Nov. 2022, pp. 1–6. doi: 10.1109/IOTSMS58070.2022.10062269.
- [41] S. Thapaliya and S. Jha, "Harnessing AI for Enhanced Identity Management: Addressing Cybersecurity Challenges in the Digital Age," presented at the The Ninth International Conference on Research in Intelligent Computing in Engineering, May 2025, pp. 71–77. doi: 10.15439/2024R34.