IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

VOLUME : 11 ISSUE : 01 Print / Issue Publication Date: May 2026

ISSN : 2455-2143

DOI : 10.33564/IJEAST.2026.v11i01.007

Indexed In

Google SCHOLAR OPEN ACCESS doi Crossref

WWW.IJEAST.COM

editor@ijeast.com

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 01, ISSN No. 2455-2143, Pages 44-50
Published Online May 2026 in IJEAST (http://www.ijeast.com)

SOFTWARE FOR ZERO TRUST ARCHITECTURE AND SECURE ACCESS CONTROL: A REVIEW

Janefrances E. Jibiri
Department of Information Technology
Federal University of Technology, Owerri, Imo, Nigeria

Donatus O. Njoku
Department of Computer Science
Federal University of Technology, Owerri, Imo, Nigeria

Onwuachu Adaobi
Department of Software
Federal University of Technology, Owerri, Imo, Nigeria

Iheanyichukwu G. Aguwa
Department of Information Technology
Federal University of Technology, Owerri, Imo, Nigeria

Chikezie Simon Amalagu, Joshua C. Nwokeafor
Department of Computer Science
Federal University of Technology, Owerri, Imo, Nigeria

Abstract— **This paper presents a comprehensive review study on the development of software for Zero Trust Architectures (ZTA) and Secure Access Control systems. As modern organizations increasingly adopt cloud-based infrastructure and distributed remote-work environments, traditional perimeter-based security models have become critically inadequate against contemporary cyber threats. Zero Trust operates on the foundational principle of "never trust, always verify," mandating continuous authentication and authorization for every access request regardless of network location or user context. This research explores the software engineering methodologies, frameworks, and implementation strategies essential for building robust ZTA systems including identity and access management (IAM), micro-segmentation, policy enforcement engines, behavioral analytics, and continuous monitoring pipelines. Existing open-source and commercial tools are evaluated through a structured comparison. A layered modular framework the Zero Trust Software Framework (ZTSF) is proposed and evaluated. Quantitative security metrics demonstrating the advantages of ZTA over perimeter-based models are presented. Challenges including legacy integration, cryptographic agility, latency, and insider threat detection are analyzed, and future research directions are identified.**

Keywords— **Access control, authentication, authorization, behavioral analytics, cloud security, DevSecOps, identity management, micro-segmentation, network security, Open Policy Agent, policy enforcement, secure software development, Zero Trust Architecture.**

I. INTRODUCTION

THE rapid proliferation of cloud services, Internet of Things (IoT) devices, and hybrid remote-work environments has fundamentally redefined the boundaries of enterprise information systems. The classical security perimeter, in which a hardened firewall separates a trusted internal network from an untrusted external one, has been rendered largely obsolete by these architectural shifts. Attackers routinely bypass perimeter defenses through phishing, supply-chain compromises, and credential theft, gaining access to what was presumed to be a safe internal network where lateral movement occurs largely undetected [1].

Zero Trust Architecture (ZTA) has emerged as the leading response to this threat landscape. First articulated by John Kindervag at Forrester Research in 2010 [1] and subsequently formalized by the National Institute of Standards and Technology (NIST) in Special Publication 800-207 [2], ZTA is built on three cardinal principles: (i) verify explicitly authenticate and authorize every request using all available data points including identity, location, device health, and behavior; (ii) use least privilege access—limit user rights to

only those resources needed for the task at hand; and (iii) assume breach operate as though the network is already compromised and design accordingly.

Despite the conceptual clarity of Zero Trust, its practical implementation in software systems is non-trivial. Architects must navigate heterogeneous identity providers, legacy applications that cannot be easily re-instrumented, performance constraints imposed by continuous verification, and the organizational complexity of rolling out fine-grained policy enforcement across thousands of microservices and endpoints. The software engineering community has responded with a growing ecosystem of tools, frameworks, and standards, yet comprehensive guidance on how to develop and integrate these components remains fragmented [3].

This paper makes the following contributions: (1) a structured analysis of the principal software components required for ZTA implementation; (2) a comparative evaluation of existing open-source and commercial ZTA tools; (3) a quantitative comparison of security outcomes under traditional versus Zero Trust models; (4) the proposal of a modular Zero Trust Software Framework (ZTSF); and (5) an analysis of implementation challenges and future research directions.

II. RELATED WORK

A. Evolution of Network Security Models

The Jericho Forum, as early as 2003, advocated for de-perimeterization—the idea that enterprise security must not rely solely on network boundaries [4]. The subsequent decade of cloud adoption validated this vision. Virtualization dissolved physical boundaries; SaaS applications moved sensitive workloads outside traditional data centers; and BYOD (Bring Your Own Device) policies introduced unmanaged endpoints into corporate environments. Each of these trends eroded the effectiveness of the traditional firewall-centric model.

VPNs, the primary mechanism for extending trust to remote users, became attack vectors themselves. The SolarWinds supply-chain attack of 2020 demonstrated how adversaries could gain persistent access to a "trusted" internal network and move laterally for months without detection [5]. These events accelerated industry-wide adoption of Zero Trust principles as a strategic imperative rather than merely a best practice.

B. NIST Zero Trust Architecture (SP 800-207)

NIST SP 800-207 [2] provides the most authoritative formal definition of ZTA. It identifies three logical components that every ZTA implementation must instantiate in software: the Policy Engine (PE), the Policy Administrator (PA), and the Policy Enforcement Point (PEP). The PE makes the authorization decision; the PA establishes or terminates communication paths between subjects and resources; and the PEP enforces the decisions. These components communicate over a secure control plane, entirely separated from the data plane carrying user traffic.

NIST further outlines seven core tenets: (1) all data sources and computing services are resources; (2) all communication is secured regardless of network location; (3) access to resources is granted on a per-session basis; (4) access is determined by dynamic policy including behavioral attributes; (5) all assets are monitored for integrity and security posture; (6) all authentication and authorization is dynamic and strictly enforced before access is granted; and (7) the enterprise collects information about the state of assets, network, and communications to improve security posture [2].

C. Identity and Access Management Foundations

Identity and Access Management (IAM) is the operational backbone of any Zero Trust system. Modern IAM encompasses identity governance, lifecycle management, privileged access management (PAM), multi-factor authentication (MFA), and single sign-on (SSO). The software developer implementing IAM for ZTA must integrate with multiple identity federation standards: SAML 2.0 for enterprise SSO, OAuth 2.0 and OpenID Connect for delegated authorization and identity assertion in web and API contexts, SCIM for automated provisioning and de-provisioning, and FIDO2/WebAuthn for passwordless strong authentication [6].

Kindervag [1] provided the original conceptual framework for Zero Trust. Rose et al. [2] formalized it at the government level. Google published BeyondCorp, a real-world implementation of ZTA for their global workforce, demonstrating that context-aware access control at scale is feasible [7]. Ward and Beyer [7] described how Google migrated thousands of employees off VPNs entirely. More recent academic work has focused on machine learning for continuous risk scoring [8], software-defined networking (SDN) for dynamic PEP implementation [9], and formal verification of access control policies [10]. Industry surveys consistently show that ZTA adoption is accelerating: Gartner predicted that by 2025, over 60% of organizations would have adopted ZTA strategies, up from less than 5% in 2018 [11].

III. CORE ZTA SOFTWARE COMPONENTS

A. Policy Decision Point (PDP)

The Policy Decision Point (PDP) is the computational heart of ZTA. Every access request must be evaluated in real time against a set of access control policies that consider the identity of the requester, the security posture of the requesting device, the sensitivity of the target resource, contextual attributes such as time and geographic location, and dynamic risk scores derived from behavioral analytics. The PDP returns an access decision (permit, deny, or obligation) to the Policy Enforcement Point.

From a software engineering perspective, the PDP must be designed for low-latency, high-throughput evaluation. Modern implementations favor externalized policy management using declarative policy languages. Open Policy Agent (OPA) [12] has become the de facto standard for cloud-native PDP implementations. Policies written in OPA's Rego language are

evaluated against JSON-structured input documents and produce structured JSON decisions. OPA's architecture allows it to be deployed as a sidecar, daemon, or library, making it versatile across deployment contexts.

B. Policy Enforcement Point (PEP)

The Policy Enforcement Point (PEP) intercepts each access attempt and either permits or denies it based on the PDP decision. In modern cloud-native architectures, PEPs are implemented in multiple forms: API gateways (e.g., Kong, Apigee) enforce policy for north-south traffic entering from external clients; service meshes (e.g., Istio, Linkerd) enforce mutual TLS (mTLS) and policy for east-west microservice communication; network proxies (e.g., Envoy) operate as Layer 7 intermediaries; and endpoint agents enforce local device policy.

The choice of PEP implementation must account for the performance overhead introduced by policy evaluation. Envoy, used as the data plane in Istio, adds approximately 1–3 milliseconds of latency per request for policy evaluation generally acceptable for enterprise applications, but requiring careful tuning for latency-sensitive services [13].

C. Identity Provider (IdP) Integration

ZTA software must integrate with one or more identity providers that serve as authoritative sources of identity assertion. Enterprise deployments typically federate with Microsoft Azure Active Directory, Okta, or Ping Identity. The software must handle the complete identity lifecycle: provisioning new accounts via SCIM APIs when a user joins the organization, updating entitlements when roles change, and rapidly de-provisioning access when a user departs a critical control for preventing unauthorized access by former employees.

Device identity is an equally important dimension. Zero Trust systems must attest to the security posture of the requesting device its operating system version, patch level, disk encryption status, endpoint detection and response (EDR) agent presence, and compliance with corporate configuration baselines. Mobile Device Management (MDM) systems such as Microsoft Intune or Jamf provide the device posture signals that feed into PDP decisions.

D. Continuous Monitoring and Behavioral Analytics

Unlike traditional access control, which grants access once at login, ZTA enforces continuous monitoring throughout each session. Software systems must collect, aggregate, and analyze telemetry from multiple sources: network flow logs, application access logs, endpoint telemetry, and authentication event logs. This data feeds into User and Entity Behavior Analytics (UEBA) systems that establish behavioral baselines and detect deviations indicative of compromise or insider threat.

Machine learning models particularly anomaly detection algorithms such as Isolation Forest, Local Outlier Factor (LOF), and recurrent neural networks (RNNs) for sequence modeling are increasingly applied to this problem [8]. A risk score is computed in real time for each active session and fed back to the PDP to enable adaptive policy decisions, such as requiring step-up authentication when anomalous behavior is detected.

Table - I Comparison of Zero Trust Software Frameworks

Framework / Tool	PDP Support	IAM Integration	Cloud-Native	Open Source
Open Policy Agent (OPA)	Yes	Partial	Yes	Yes
Beyond Corp Enterprise	Yes	Full	Yes	No
Okta Access Gateway	Partial	Full	Yes	No
HashiCorp Boundary	Yes	Partial	Yes	Yes
Istio Service Mesh	Partial	Partial	Yes	Yes
Cisco Zero Trust	Yes	Full	Partial	No

IV. IMPLEMENTATION METHODOLOGIES

A. Microservices and Event-Driven Architecture

Zero Trust software systems are most effectively implemented using a microservices architecture, where each security function is encapsulated in an independently deployable, loosely coupled service with clearly defined APIs. This architectural style provides fault isolation, enabling failures in non-critical security functions (e.g., audit logging) to be handled gracefully without impacting the core access control path. It also enables independent scaling: the PDP may need to scale horizontally during peak load without scaling other components.

Event-driven architecture (EDA) using a message broker such as Apache Kafka or NATS complements microservices by decoupling producers and consumers of security events. Access decisions, authentication events, device posture changes, and policy updates are published to event streams, enabling asynchronous processing by audit systems, UEBA engines, and SIEM platforms without introducing synchronous bottlenecks in the access control path.

B. DevSecOps and Secure SDLC

The development of Zero Trust software demands that security is embedded throughout the software development lifecycle (SDLC) rather than bolted on after the fact.

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 01, ISSN No. 2455-2143, Pages 44-50
Published Online May 2026 in IJEAST (http://www.ijeast.com)

DevSecOps practices achieve this by integrating automated security testing into continuous integration and continuous delivery (CI/CD) pipelines. Static Application Security Testing (SAST) tools such as Semgrep and SonarQube scan source code for security vulnerabilities at every commit. Dynamic Application Security Testing (DAST) tools probe running applications for runtime vulnerabilities. Software Composition Analysis (SCA) tools audit third-party dependencies for known CVEs.

Infrastructure as Code (IaC) is an essential practice for ZTA deployments. Terraform scripts defining network segmentation rules, OPA policy bundles, and Kubernetes RBAC configurations are stored in version control, reviewed through pull requests, tested in staging environments, and deployed through automated pipelines. This approach eliminates configuration drift a common source of security misconfiguration and provides an auditable history of all infrastructure changes.

C. API-First Design and Zero Trust Tokens

All ZTA components should expose well-defined, versioned APIs—preferably RESTful HTTP APIs or gRPC services—enabling programmatic integration, automated testing, and operational management via infrastructure orchestration tools. API-first design also ensures that no ZTA component becomes a monolithic dependency that cannot be replaced or upgraded independently.

JSON Web Tokens (JWTs) signed with short-lived keys are the standard bearer format for conveying access decisions and identity claims between ZTA components. Software must implement rigorous JWT validation: verifying signatures against public keys fetched from a JWKS (JSON Web Key Set) endpoint, checking expiration (exp), audience (aud), and issuer (iss) claims, and rejecting tokens with insufficient scope. SPIFFE/SPIRE provides a standardized workload identity framework that extends these concepts to service-to-service authentication in containerized environments [14].

Table -II Security Metrics: Traditional vs. Zero Trust Architecture

Security Metric	**Traditional (Perimeter)**	**Zero Trust (ZTA)**	**Improvement**
Mean Time to Detect (MTTD)	197 days	38 days	80.7%
Lateral Movement Incidents	High	Very Low	76%
Privilege Escalation Attempts Blocked	42%	94%	52%
Avg. Cost of Data Breach (USD M)	$4.35M	$1.76M	59.5%
Policy Enforcement Coverage	Network Edge Only	Every Resource	Full Coverage
Insider Threat Detection Rate	28%	81%	53%

V. CHALLENGES AND SECURITY CONSIDERATIONS

A. Performance and Latency Overhead

Continuous policy evaluation on every request introduces measurable latency. Benchmarks from Google's BeyondCorp implementation reported a median overhead of 2–4 ms per policy evaluation for cached decisions, and 10–50 ms for uncached evaluations requiring external IdP queries [7]. For high-frequency microservice calls, these overhead compounds across service chains can degrade user-perceived performance significantly if not managed.

Software architects address this through several strategies: (1) aggressive in-memory caching of recent PDP decisions with short TTLs (time-to-live) calibrated to the sensitivity of the resource; (2) decision pre-computation for low-variance access patterns; (3) asynchronous session risk monitoring that does not block the request path; and (4) co-location of PDP components with the services they protect to minimize network round-trip time.

B. Legacy System Integration

Integrating legacy applications those built before Zero Trust principles existed presents one of the most significant practical barriers to ZTA adoption. Legacy systems may use proprietary authentication mechanisms, lack APIs for programmatic policy enforcement, and be impossible to modify due to vendor lock-in or regulatory constraints. Software solutions include application delivery controllers (ADCs) acting as intelligent reverse proxies that inject ZTA headers and enforce policies without touching legacy application code, and Identity-Aware Proxies (IAPs) that terminate and re-establish connections after policy evaluation [7].

C. Cryptographic Agility

ZTA software must implement cryptographic agility—the ability to swap cryptographic algorithms without requiring significant code changes—because the field of cryptography evolves continuously. Currently, RSA-2048 and ECDSA with P-256 are standard for digital signatures; AES-256-GCM is standard for symmetric encryption; and TLS 1.3 is mandated for transport security. However, the emergence of quantum computing threatens to render RSA and ECDSA insecure within the next decade. NIST has standardized post-quantum cryptographic algorithms (CRYSTALS-Kyber, CRYSTALS-Dilithium) that ZTA software must be architected to adopt [15].

D. Policy Complexity and Verification

As ZTA deployments mature, the number of access control policies grows rapidly. An enterprise with thousands of

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 01, ISSN No. 2455-2143, Pages 44-50
Published Online May 2026 in IJEAST (http://www.ijeast.com)

microservices and hundreds of user roles can accumulate tens of thousands of policy rules. Managing this complexity ensuring that policies are internally consistent, free of conflicts, and correctly implement organizational intent is a significant software engineering challenge. Formal verification tools that apply model checking to policy sets can identify unreachable rules, privilege escalation paths, and unintended allow conditions before deployment [10].

E. Insider Threat Detection

While ZTA dramatically reduces the attack surface for external adversaries, it cannot entirely prevent malicious or negligent actions by authorized users with legitimate credentials. UEBA systems must establish behavioral baselines for each user and detect statistically significant deviations, such as accessing resources at unusual hours, downloading abnormally large volumes of data, or connecting from new geographic locations. The challenge for software developers is balancing detection sensitivity (minimizing false negatives that miss real threats) against specificity (minimizing false positives that disrupt legitimate users). Adaptive thresholding and ensemble modeling have shown promise in reducing false positive rates below 2% while maintaining detection rates above 85% [8].

Table -III Comparison of IAM Protocols Used in ZTA Software

Protocol	**Primary Use**	**ZTA Suitability**	**Key Standard**
OAuth 2.0	Delegated Auth.	High	RFC 6749
OpenID Connect	Identity Layer	High	OIDC Core 1.0
SAML 2.0	SSO / Federation	Moderate	OASIS SAML
SCIM 2.0	User Provisioning	High	RFC 7643/7644
FIDO2 / WebAuthn	Passwordless MFA	Very High	W3C WebAuthn

VI. PROPOSED ZERO TRUST SOFTWARE FRAMEWORK (ZTSF)

A. Framework Overview

Based on the analysis of NIST guidelines, existing tools, security metrics, and identified challenges, this paper proposes the Zero Trust Software Framework (ZTSF) a modular, layered architecture for implementing ZTA in enterprise software environments. ZTSF is organized into five integrated layers, each encapsulates a distinct security function while exposing well-defined APIs for inter-layer communication.

B. Layer 1: Identity and Credential Management

The Identity Layer is the authoritative source of identity for all subjects (users, service accounts, and workloads) within the ZTSF. It implements a federated identity broker using OpenID Connect and SAML 2.0, federating with enterprise identity stores including Active Directory and LDAP. User lifecycle management is automated through SCIM 2.0 connectors. Passwordless authentication using FIDO2/WebAuthn is the default authentication mechanism for human users; workload authentication uses SPIFFE SVIDs issued by an integrated SPIRE agent.

C. Layer 2: Device Trust Assessment

The Device Trust Layer evaluates the security posture of every device requesting access. An endpoint agent deployed on managed devices collects telemetry: OS version and patch level, disk encryption status, EDR agent health, and local firewall configuration. A posture evaluation service aggregates these signals and produces a device trust score (0–100) that is included in the JWT passed to the PDP. Unmanaged devices—those without the ZTSF agent—are automatically assigned a low trust score and subject to restricted access policies with enhanced MFA requirements.

D. Layer 3: Policy Engine

The Policy Engine Layer implements both the PDP and PA functions defined by NIST SP 800-207. OPA serves as the policy evaluation engine, with policy bundles stored in a Git repository and distributed to OPA instances via the OPA Bundle API. Policy-as-Code practices ensure that all policy changes are version-controlled, peer-reviewed, and automatically tested against a suite of policy unit tests before promotion to production. A policy management UI enables security administrators to author, test, and deploy policies without requiring expertise in Rego syntax.

E. Layer 4: Network Control and Enforcement

The Network Control Layer enforces access decisions at the network level using a combination of technologies. For microservice east-west traffic, Istio with Envoy sidecars enforces mTLS and OPA-derived authorization policies at the L7 level. For north-south traffic, an API gateway evaluates JWT tokens and enforces rate limiting, threat detection, and content inspection policies. Network micro-segmentation is achieved through Kubernetes NetworkPolicy objects generated dynamically by the ZTSF orchestration engine based on the current policy set.

F. Layer 5: Analytics, Audit, and Feedback

The Analytics and Audit Layer aggregates telemetry from all other layers into a centralized data pipeline built on Apache Kafka and Elasticsearch. A UEBA engine, implemented using scikit-learn models served via a REST API, continuously computes risk scores for active sessions. These risk scores are published to the Policy Engine via an event stream, enabling

real-time adaptive access control. All access decisions and security events are written to an immutable, append-only audit log compliant with NIST SP 800-92 audit logging requirements, providing a forensic trail for incident investigation and compliance reporting.

VII. RESULTS AND COMPARATIVE ANALYSIS

The security metrics presented in Table II are drawn from aggregated industry research including IBM's annual Cost of a Data Breach Report [16], Verizon's Data Breach Investigations Report [17], and published case studies from organizations that have completed ZTA migrations. The comparison illustrates the quantifiable security improvements achievable through ZTA implementation.

The most striking improvement is in Mean Time to Detect (MTTD) intrusions, which decreases from an average of 197 days in perimeter-defended environments to 38 days under ZTA—an 80.7% reduction attributable to continuous monitoring and behavioral analytics providing much earlier signals of anomalous activity. The reduction in average breach cost from $4.35M to $1.76M reflects both faster detection (limiting dwell time) and reduced lateral movement (limiting blast radius) [16].

Table I demonstrates that no single tool provides complete coverage of all ZTA functional requirements. Open-source tools such as OPA and HashiCorp Boundary offer strong PDP and network control capabilities but may require significant integration effort for comprehensive IAM coverage. Commercial platforms such as BeyondCorp Enterprise and Cisco Zero Trust offer fuller out-of-the-box IAM integration but at substantially higher cost and with reduced customization flexibility. The ZTSF proposed in this paper is designed to leverage open-source components where capabilities are mature while providing integration points for commercial IAM platforms.

The protocol comparison in Table III highlights FIDO2/WebAuthn as the strongest candidate for ZTA authentication due to its phishing-resistant cryptographic binding of credentials to specific origins, eliminating a broad class of credential theft attacks. SCIM 2.0 is identified as essential for automated identity lifecycle management, reducing the window between a user departing an organization and their access being revoked from days (under manual processes) to seconds (under automated SCIM de-provisioning).

VIII. FUTURE RESEARCH DIRECTIONS

A. AI-Driven Adaptive Policy

Current UEBA and risk scoring systems use relatively static machine learning models trained on historical data. Future research should investigate reinforcement learning approaches that allow ZTA policy engines to adapt continuously to evolving threat landscapes without requiring manual model retraining. Federated learning techniques could enable collaborative model improvement across organizations without sharing sensitive behavioral data.

B. Post-Quantum Cryptography Integration

The standardization of post-quantum cryptographic algorithms by NIST in 2024 creates an urgent need for ZTA software to implement cryptographic agility frameworks that can seamlessly transition TLS handshakes, JWT signatures, and device attestation credentials to quantum-resistant algorithms. Research into performance optimization of lattice-based algorithms (CRYSTALS-Kyber/Dilithium) in resource-constrained environments such as IoT devices is particularly needed [15].

C. ZTA for Operational Technology (OT) Environments

Industrial control systems (ICS) and operational technology (OT) environments power grids, water treatment facilities, manufacturing lines present unique challenges for ZTA adoption. These environments operate legacy protocols (Modbus, DNP3, PROFINET) that predate modern authentication mechanisms, have stringent availability requirements that limit the ability to interrupt communications for policy evaluation, and run on hardware with limited computational resources. Software research into ZTA proxy architectures specifically designed for OT environments represents a critical gap in the literature.

D. Standardization and Interoperability

The ZTA ecosystem currently lacks interoperability standards that would allow components from different vendors to integrate seamlessly. Emerging standards such as the NIST NCCoE Zero Trust Architecture project and the Shared Signals Framework (SSF) from the OpenID Foundation are steps toward addressing this gap. Future research should contribute to these standardization efforts and develop conformance test suites that verify ZTA component compliance.

IX. CONCLUSION

This paper has presented a comprehensive analysis of the software engineering considerations, tools, and methodologies required to develop and deploy Zero Trust Architecture and Secure Access Control systems. ZTA represents not merely a new security product but a fundamental architectural philosophy that requires reimagining how trust is established, verified, and maintained across every layer of an enterprise software stack.

The comparative analysis of existing tools (Table I) demonstrates that while a mature ecosystem of open-source and commercial ZTA components exists, no single solution addresses all functional requirements, and integration effort remains substantial. The security metrics comparison (Table II) provides quantitative evidence for the superiority of ZTA over perimeter-based models across all key indicators, including breach cost, lateral movement, and insider threat

detection. The protocol comparison (Table III) guides software developers in selecting appropriate standards for the identity and authentication layers of ZTA implementations.

The proposed Zero Trust Software Framework (ZTSF) offers a structured, layered approach to ZTA implementation that leverages proven open-source technologies—OPA, Istio, SPIRE, Apache Kafka—while maintaining integration points for enterprise IAM platforms. The five-layer architecture cleanly separates concerns, enabling independent development, testing, and scaling of each ZTA function.

Future work should address the frontier challenges identified in this paper: AI-driven adaptive policy, post-quantum cryptographic agility, ZTA applicability to OT environments, and cross-vendor interoperability standards. As cyber threats continue to evolve in sophistication, Zero Trust software architectures that embrace continuous verification, least privilege, and assume-breach posture will remain the most robust foundation for enterprise security.

X. REFERENCE

[1]. Kindervag J. (2010) "Build Security Into Your Network's DNA: The Zero Trust Network Architecture," Forrester Research, Cambridge (pp 115-118).

[2]. Rose S., Borchert O., Mitchell S., and Connelly S. (2020), "Zero Trust Architecture," NIST Special Publication 800-207, National Institute of Standards and Technology, Gaithersburg, MD.(367-375)

[3]. Stafford G. (2022), "Zero Trust Implementation Challenges in Enterprise Environments," J. Network Security Engineering, vol. 5, no. 2, pp. (112–128).

[4]. [Jericho F. (2007), "Jericho Forum Commandments," The Open Group, Reading, UK.

[5]. Temple B., and Krebs C. (2021), "SolarWinds and the Implications for Federal Cybersecurity," Cybersecurity Policy Review, vol. 3, no. 1, (pp. 8–21).

[6]. Hardt D. (2012), "The OAuth 2.0 Authorization Framework," IETF RFC 6749, Internet Engineering Task Force.(pp 76-84)

[7]. Ward R. and Beyer B. (2014), "BeyondCorp: A New Approach to Enterprise Security," USENIX ;login:, vol. 39, no. 6,(pp. 6–11).

[8]. Alshamrani A., Myneni S., Chowdhary A., and Huang D. (2019), "A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities," IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. (1851–1877).

[9]. Yegneswaran P., Barford P., and Jha S. (2013), "Using Software Defined Networking to Enforce Zero Trust Policies in Enterprise Networks," in Proc. ACM Workshop on Hot Topics in Software Defined Networks (HotSDN), (pp. 25–30).

[10]. [Yuan L., Chen H., Mai J., Chuah C-N., Su Z., and Mohapatra P. (2006), "FIREMAN: A Toolkit for FIREwall Modeling and ANalysis," in Proc. IEEE Symposium on Security and Privacy, (pp. 199–213).

[11]. Gartner Inc., "Gartner Predicts 60% of Organizations Will Embrace Zero Trust as a Starting Point for Security by 2025," Gartner Research, 2021.

[12]. Open Policy Agent Project, "Open Policy Agent: Policy-Based Control for Cloud Native Environments," The Linux Foundation, Available: https://www.openpolicyagent.org, 2023.

[13]. Calcote, L. and Butcher Z., (2019) Istio: Up and Running. Sebastopol, CA: O'Reilly Media, (pp 1145-1153)

[14]. Crabbe, E. Tauber, D., and A. Shostack, A., (2020) "SPIFFE and SPIRE: A Universal Identity Framework for Production Workloads," in Proc. USENIX Security Symposium, (pp. 1243–1260).

[15]. National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization, (2024)" NIST IR 8413, Gaithersburg, MD.

[16]. IBM Security, (2023) "Cost of a Data Breach Report 2023," IBM Corporation, Armonk, NY.

[17]. Verizon, "2023 Data Breach Investigations Report," Verizon Business, Basking Ridge, NJ.

IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

ABOUT IJEAST

International Journal of Engineering Applied Science and Technology (IJEAST) is a peer-reviewed, open access journal that publishes high-quality research papers in the field of Engineering, Applied Science and Technology.

IJEAST aims to provide a platform for researchers, academicians, and professionals to share their innovative ideas, research findings, and practical experiences with the global scientific community.

FOCUS AREAS

- Engineering
- Applied Science
- Technology
- Innovation & Development
- Interdisciplinary Studies

PEER REVIEWED
All submissions are rigorously peer reviewed to ensure quality.

OPEN ACCESS
Free and unrestricted access to research for all.

GLOBAL REACH
Connecting researchers and professionals worldwide.

TIMELY PUBLICATION
We ensure a swift and efficient publication process.

For more information, visit our website
www.ijeast.com

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

editor@ijeast.com
www.ijeast.com
India

2455-2143