Quantitative Graph-Based Security Domain Segmentation for Corporate Network Risk Reduction

Sherzod Sayfullaev[1], Abdukhalil Ganiev[1], Sergei Kesel[2] and Nargiza Sayfullaeva[1]
[1]*Information Security Department, Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Amir Temur Str. 108, 100084 Tashkent, Uzbekistan*
[2]*Moscow Polytechnic University, Bolshaya Semyonovskaya Str. 38, 107023 Moscow, Russia*
sherzodsay@gmail.com, ganiyevabduhalil96@gmail.com, sakesel161@gmail.com, n.sayfullaeva07@gmail.com

Keywords: Corporate Network Security, Graph Partitioning, Security Domains, Risk Scoring, Network Segmentation, Zero Trust, SDN.

Abstract: This paper proposes a graph-based method for forming corporate-network security domains from object-level risk attributes. Importance, vulnerability, attack probability, and estimated loss are combined into a continuous risk score and mapped to discrete security levels. Objects with equal levels are grouped into disjoint domains, mixed-level service conflicts are resolved, and boundary controls are inserted on inter-domain edges. A Python prototype implements the scoring, graph partitioning, and boundary-insertion sequence. The method is evaluated on a synthetic 150-node corporate topology with 5,000 daily flows. Relative to a flat baseline, the reported scenario reduces average blast radius from 100% to 18.5% and directed attack paths from 22,350 to 1,240. Unlike VLANs, which provide logical network separation, Zero Trust, which applies per-request policy decisions, and SDN, which provides programmable forwarding control, the proposed method computes risk-based domain membership and boundary locations. It is therefore complementary to these enforcement approaches. The study is a simulation and prototype evaluation rather than a production-network deployment.

1 INTRODUCTION

Enterprise networks combine workstations, servers, gateways, and services with different exposure and business criticality. Segmentation is used to constrain lateral communication and isolate assets, but existing technologies solve different parts of the problem. IEEE 802.1Q provides logical separation through VLAN mechanisms [1]. Zero Trust Architecture treats network location as insufficient for trust and relies on explicit policy decisions and continuous verification [2]. SDN provides programmable control over forwarding behavior through a logically centralized control framework [3].

These mechanisms provide isolation or policy enforcement, but they do not by themselves define a risk-scoring rule for assigning heterogeneous assets to domains. The question addressed here is: given a network graph and object-level importance, vulnerability, exposure, and loss indicators, how should objects be grouped into non-overlapping security domains and where should boundary controls be inserted? Vulnerability severity may be represented using standardized scoring inputs such as CVSS [4].

The study contributes a continuous object-level risk score, discrete security-level mapping, a graph algorithm with mixed-service conflict resolution and boundary insertion, a Python prototype, and a simulation-based before/after evaluation. The method does not replace VLAN, Zero Trust, or SDN; it computes domain assignments and boundary locations that can be translated into those enforcement mechanisms.

2 GRAPH-BASED SECURITY DOMAIN SEGMENTATION METHOD

In this study, a security domain is a non-overlapping set of network objects assigned the same security level. A security object may be a computer, server, router, gateway, or service instance. The level is

computed from object attributes and used as the grouping criterion.

Domain boundaries are difficult to define when assets with different criticality share hosts, services, or communication paths. Overlap can preserve lateral movement between high- and low-risk objects; therefore, the proposed method requires disjoint domain membership after conflict resolution.

Distributed client-server applications illustrate this problem because communicating components may be geographically separated while still requiring controlled trust relationships. Domain formation is therefore based on object risk and communication topology rather than physical proximity.

The hierarchy can be visualized as concentric security levels (Fig. 1). The outer region represents lower sensitivity and the inner region higher sensitivity. In the proposed method, actual level assignment is determined by the risk score and threshold mapping in Section 2.3.

Figure 1 is a conceptual visualization only; operational domain membership is produced by the graph algorithm and object attributes.

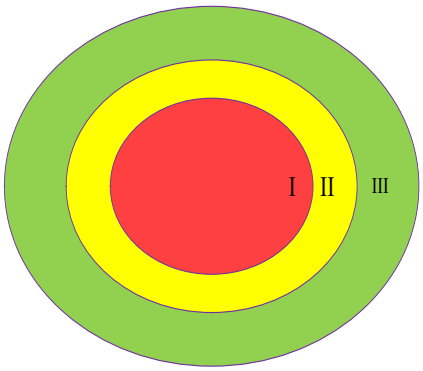

Figure 1: Hierarchy of security levels in the form of concentric rings.

Traffic exposure provides a practical motivation for segmentation. A public-facing service and a high-value internal system should not remain in the same unrestricted communication zone. Separating them reduces the set of reachable objects and creates a defined point for access-control enforcement.

The method therefore groups objects by computed security requirements and preserves only policy-permitted communication between domains.

The procedure is a graph transformation: assign a security level to each object, group equal-level objects, resolve mixed-level services, and insert boundary objects on edges that cross domain boundaries.

The network is represented as G=(V,E), where vertices are protected objects and edges are communication relationships. Graph-based cybersecurity assessment is suited to representing attack and reachability relationships [5]. The output is a domain assignment for V and a set of boundary edges requiring control.

2.1 Separation of Security Domains

The separation stage converts object attributes into security levels through the following sequence:

- Identification of network objects that require protection (computers, servers, routers);
- Assignment of an importance level to each object within the network structure for subsequent evaluation of its security level;
- Identification of potential attack types applicable to each object;
- Determination of vulnerability classes that may enable or facilitate the identified attacks for each object;
- Estimation of the probability of the most common vulnerability classes;
- Calculation of potential losses from a specific attack by comparing vulnerabilities and their probabilities while considering the importance level.

By completing these steps, it becomes possible to conclude which security level each security object belongs to. As a result, sets of security objects grouped by security level are obtained.

Subsequently, objects with identical security levels are combined into a single network, forming security domains. That is, the graph is partitioned into subgraphs, where each subgraph contains security objects of the same level.

Situations may arise in which two or more objects require different security levels but are located within the same domain, which is unacceptable.

When a host runs services with different security requirements, a single host-level domain can hide a security conflict. The algorithm detects mixed-level service sets and represents them as separate virtual or physical service vertices before clustering, so each service inherits its own security level.

2.2 Integration of Security Domains

After separation, disjoint domains are reintegrated into one logical network graph. Edges between different domains are replaced by boundary-control objects, preserving required communication as a controlled path rather than an unrestricted direct edge.

Inter-domain communication is enforced through gateways, firewalls, or equivalent policy components. The algorithm identifies boundary edges and source/destination security levels rather than prescribing a vendor-specific enforcement plane. This separation is consistent with network-level security-service integration approaches [6].

Boundary-control events should be logged so that denied or bypass attempts can be investigated and used to update threat-probability inputs.

Incorrect grouping or unrestricted integration increases cross-level communication paths and may degrade confidentiality, integrity, and availability. The method therefore evaluates structural partitioning and inter-domain access control together.

Figure 2 illustrates the conceptual problem of overlapping security domains.

Confidentiality, integrity, and availability are treated jointly. Figures 3 and 4 illustrate an inter-object edge before and after insertion of a boundary-control object.

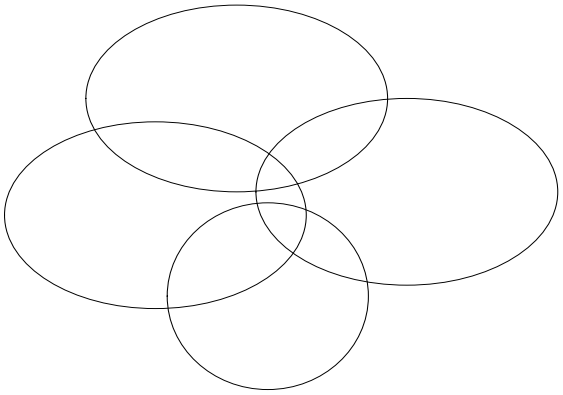

Figure 2: Overlapping security rings.

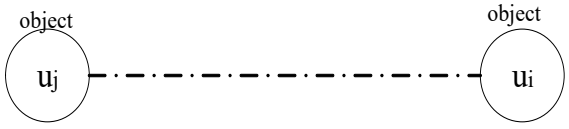

Figure 3: Connection between objects before inserting a boundary object.

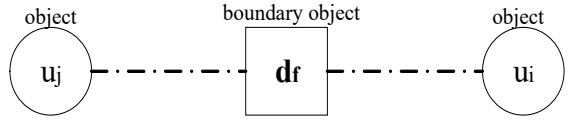

Figure 4: Connection between objects after inserting a boundary object.

2.3 Formal Mathematical Model for Security Level Computation

Let the corporate network be defined as a set of objects $O = \{o_1, o_2, \dots, o_n\}$. For each object $o_n \in O$, we define the following input parameters:

- $I(o_i) \in [0,1]$: The normalized *Importance Level of* the object (e.g., 1.0 for a core database, 0.2 for a guest printer).
- $V(o_i) \in [0,1]$V(v): The Vulnerability Score. The value may be derived from CVSS severity components or another documented enterprise scoring scheme [4].
- $P_a(o_i) \in [0,1]$: The *Probability of Attack*, estimated based on historical threat intelligence and exposure (e.g., external-facing vs. internal).
- $L(o_i) \in R^+$: The *Estimated Financial or Operational Loss* incurred if the object is compromised.

Step 1.1: Compute the Continuous Risk Score

The continuous risk score $R(o_i)$ for each object is calculated as the product of the probability of a successful exploit and the weighted impact:

$$R(o_i) = P_a(o_i) \cdot V(o_i) \cdot L(o_i) \cdot I(o_i) \quad (1)$$

Step 1.2: Map to Discrete Security Levels

To group objects into domains, the continuous risk score must be mapped to a discrete Security Level $SL(o_i) \in Z^+$. We define threshold values $\tau_1, \tau_2, \dots, \tau_{k-1}$ to categorize the risk into *k* distinct levels. For a standard 3-tier architecture (Low, Medium, High):

$$SL(o_i) = \begin{cases} 1, if\ R(o_i) \le \tau_1 \\ 2, if\ \tau_1 < R(o_i) \le \tau_2 \\ 3, if\ R(o_i) > \tau_2 \end{cases} \quad (2)$$

Once thresholds are fixed, the mapping from object attributes to a discrete security level is deterministic.

2.4 Graph Partitioning and Boundary Insertion Algorithm

We represent the network as a graph $G = (V, E)$, where *V* represents network objects and *E* represents communication links.

The goal is to partition *G* into a set of disjoint subgraphs (security domains) $D = \{D_1, D_2, \dots, D_m\}$ where all vertices in a given D_j share the same *SL*.

Here is the pseudocode for the domain formation and conflict resolution process:

```
Input: Network Graph G = (V, E), Threat Data (Vulnerabilities, Probabilities, Importance, Loss)
Output: Set of isolated Security Domains D, Updated Graph G' with Boundary Mechanisms
1: Initialize empty set of domains D
   2: FOR EACH vertex v in V DO
   3:  // Calculate Risk and Security Level
   4:  R(v) = P_a(v) * V(v) * L(v) * I(v)
   5:  SL(v) = MapToDiscreteLevel(R(v))
   6: END FOR
   7: // Conflict Resolution: Split nodes running services with mixed security requirements
   8: FOR EACH vertex v in V DO
   9:  IF v hosts multiple services (s_1, s_2, ..., s_k) DO
   10:  IF exists i, j such that SL(s_i) != SL(s_j) THEN
   11:  // Rule: Services with different levels must not be combined in a single system
   12:  Split v into new virtual/physical vertices v_i, v_j
   13:  Assign SL(v_i) = SL(s_i) and SL(v_j) = SL(s_j)
   14:  Update G with new vertices and redirect corresponding edges E
   15:  END IF
   16:  END IF
   17: END FOR
   18:  // Domain Clustering (Grouping identical security requirements)
   19: FOR EACH unique security level L in {1, 2, ..., k} DO
   20:  Create new Domain D_L
   21:  FOR EACH vertex v in V DO
   22:  IF SL(v) == L THEN
   23:  Add v to D_L
   24:  END IF
   25:  END FOR
   26:  Add D_L to D
   27: END FOR
   28:  // Domain Integration (Inserting boundaries)
   29: FOR EACH edge e(u, w) in E DO
   30:  IF u is in D_i AND w is in D_j AND i != j THEN
   31:  Remove edge e(u, w)
   32:  Insert Boundary_Object B (e.g., Firewall/Gateway) between u and w
   33:  Add edges e(u, B) and e(B, w) to G'
   34: Apply Access_Control_Policy(B, SL(u), SL(w))
   35:  END IF
   36: END FOR
37: RETURN D, G'
```

Algorithm 1: Security Domain Formation and Conflict Resolution.

Algorithm 1 operationalizes two constraints: objects are clustered by computed security level, and services with different levels are not forced into one domain. Inter-domain edges are then converted into controlled paths through boundary objects.

2.5 Prototype Software Implementation

A Python/NetworkX prototype executes Algorithm 1. Network objects are stored as nodes with risk attributes; the program computes R(v), maps scores to security levels, groups nodes by level, and replaces cross-domain edges with boundary-control nodes. It also computes blast radius and policy-permitted directed attack-path counts.

The prototype is a research implementation rather than a production firewall, Zero Trust engine, or SDN controller. It outputs domain assignments and boundary locations. The accompanying Python source reproduces the algorithmic sequence.

3 COMPUTABLE METRICS FOR OVERALL NETWORK SECURITY LEVEL (ONSL)

The Overall Network Security Level (ONSL) is a study-specific comparison index combining the Structural Segmentation Index (SSI) and Inter-Domain Access Control (IAC) efficiency. Related security-analysis systems use computable indicators for network assessment [7]. More broadly, computational modeling can transform measured operational variables into quantitative engineering indicators suitable for structured analysis [8], while data-driven risk-matrix workflows convert multi-factor inputs into structured priorities [9]. Here, ONSL is used only for before/after comparison of the same simulated topology.

The general formula is defined as:

$$ONSL = \alpha \cdot SSI + \beta \cdot IAC \tag{3}$$

where α and β are the importance weights of the network structure and boundary protection mechanisms, respectively, satisfying the condition $\alpha + \beta = 1$.

3.1 Structural Segmentation Index (SSI)

This component quantifies how effectively the network is partitioned into domains. Let N_{total} be the total number of protected objects (services) in the network, and D be the set of security domains.

$$SSI = \frac{|D|}{max_{k\in\{1\ldots|D|\}}(S_k)\times N_{total}}. \tag{4}$$

where:

- $|D|$ is the total number of security domains in the network.
- S_k is the number of services or objects in the *k*-th domain.
- $max_k(S_k)$ represents the maximum number of services located in any single domain.

Logical Proof: In the ideal secure network state proposed in the paper, each domain contains only a single service. In this scenario ($|D| = N_{total}$ and $max_k(S_k) = 1$), the *SSI* evaluates to 1. In the worst-case scenario, all services are located within a single domain. Here ($|D| = 1$ and $max_k(S_k) = N_{total}$), the $SSI = \frac{1}{(N_{total})^2} \approx 0$. This establishes the limiting behavior of the SSI metric, and the resulting relationship is illustrated in Figure 6.

3.2 Inter-Domain Access Control (IAC)

The IAC metric measures the effectiveness of the boundary protection objects (e.g., gateways and firewall systems) placed between domains.

$$IAC = \frac{1}{|E_b|}\sum_{e\epsilon E_b} W(e), \quad (5)$$

where:

- E_b is the set of logical or physical communication links (edges) connecting different domains.
- $W(e) \in [0,1]$ is the security effectiveness score of the firewall rules on link *e* (e.g., a strict "Deny All" policy might equal 1.0, partially open ports might equal 0.5, and unrestricted access equals 0).

3.3 Delta Calculation (Before and After Segmentation)

To evaluate the effectiveness of the method, we analyze the changes in the overall network security level before and after domain separation and integration. We define this change as $\Delta ONSL$:

$$\Delta ONSL = ONSL_{finish} - ONSL_{start} \quad (6)$$

- *Before Segmentation* ($ONSL_{start}$): The network has not been partitioned ($|D|=1$), and there are no internal boundary protection mechanisms ($IAC = 0$). Consequently:

$$ONSL_{start} = \left(\frac{1}{N_{total}^2}\right) + 0 \approx 0. \quad (7)$$

- *After Segmentation* ($ONSL_{finish}$): The network objects are divided into *m* domains based on their security levels, and boundary mechanisms are inserted. Consequently:

$$ONSL_{finish} = \alpha\left(\frac{m}{max_k(S_k)\cdot N_{total}}\right) + \beta(IAC_{new}). \quad (8)$$

$\Delta ONSL > 0$A positive difference indicates improvement in the study-specific ONSL index for the same scenario; it is not presented as a universal security certification score.

4 CORPORATE NETWORK SECURITY ENHANCEMENT METHOD

Effectiveness is evaluated by comparing the same topology before and after graph-based domain formation. The initial and transformed structures are evaluated with the same metrics.

Figure 5 shows the corporate-network topology used to illustrate the evaluation procedure.

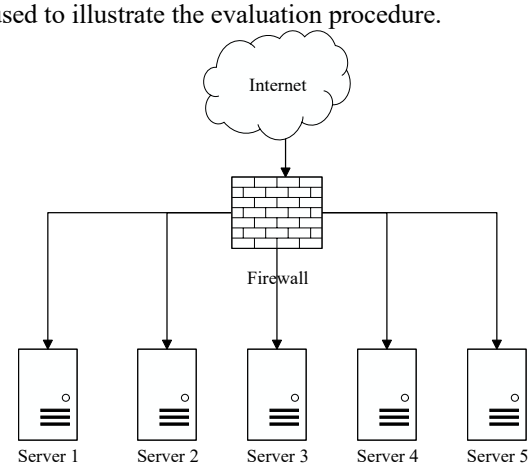

Figure 5: Example of a corporate network for effectiveness evaluation.

$D_{start}D_{finish}$Let D0 and D1 denote the domain sets before and after segmentation, with vectors describing the services or characteristics assigned to each domain.

$$\overline{T_{start}T_{finish}T_{start}T_{finish}T_{start}}$$

Equation (9) compares the aggregate characteristics of the initial and resulting domain structures.

$$\left||\overline{T_{start}}|\right| = \left|\left|\overline{T_{finish}}\right|\right|, but\ |\overline{T_{start}}| \neq |\overline{T_{finish}}|. \quad (9)$$

$\|V_0\|\|V\|$Two boundary cases are used for interpretation: one protected service per domain and all services in one domain.

These cases provide reference points for interpreting the segmentation index.

Equation (10) expresses the ordering between the maximum- and minimum-isolation states.

$$\left|\overline{T_{finish_j}}\right| = 1, \forall\, j \in [1, D_{finish}]. \quad (10)$$

$D_{start} \to max$ and $\left|\overline{T_{finish_j}}\right| \to 1$ at the same time.

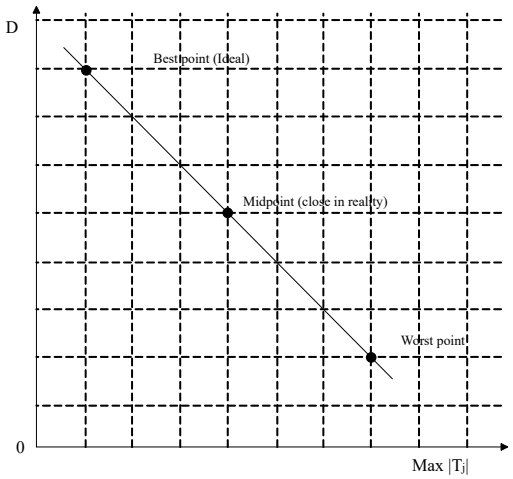

Figure 6: Dependence of the security level on the ratio of the number of domains to the maximum number of objects per domain.

For SSI, increasing domain count while reducing the maximum domain size improves structural isolation. This is not sufficient alone; effective inter-domain policies are represented by IAC.

Figure 6 visualizes the SSI relationship between domain count and maximum domain size.

5 SIMULATION-BASED EVALUATION AND COMPARISON

The method was evaluated on a synthetic 150-node topology containing 120 workstations, 10 DMZ servers, and 20 core servers with 5,000 daily source-destination flows. The flat baseline had no internal boundary controls; the segmented topology used DMZ, Internal, and Core domains. The Python prototype executed risk-level mapping and graph transformation. This is a simulation, not a production deployment.

Blast Radius (BR) was defined as the percentage of nodes reachable from a selected infected host under the active policy set. Attack paths were counted as policy-permitted directed node pairs:

$$BR(v) = \frac{|Reach(v)|}{|V|} \times 100\%.$$

$Reach(v)$where the numerator is the number of nodes reachable from the infected host and |V|-1 is the maximum number of other nodes reachable in a flat network.

Table 1 summarizes the before/after results. BR and attack-path counts are topology-derived graph metrics. CIA values are analyst-assigned ordinal scenario scores on a 1–10 scale and are included as descriptive indicators rather than statistically measured security outcomes.

Table 1: Measurable metrics before and after segmentation.

Metric	Flat Network (Baseline)	Segmented Network (3 Domains)	Impact / Change
Average Blast Radius (BR)	100%	18.5%	-81.5% (Damage contained)
Attack Paths	22,350	1,240	-94.4% (Reduced attack surface)
CIA Score (1-10 scale)	C: 2, I: 3, A: 4	C: 9, I: 8, A: 8	~2.5x improvement
Firewall Policy Rules	12	345	+2775%

The simulation shows containment and reachability effects for the defined topology. VLANs use administrator-defined logical membership [1], Zero Trust applies explicit policy decisions [2], and SDN provides programmable forwarding control [3]. The proposed method differs by computing the recommended partition from risk scores; no universal numerical superiority over those architectures is claimed. Recent studies on Zero Trust automation and orchestration [10] and Zero Trust policy integration in programmable 6G O-RAN environments [11] further indicate that future segmentation systems should separate risk-driven domain formation from dynamic policy enforcement. Secure link-level communication technologies, including LiFi-based IoT sensor networks [12], can complement boundary enforcement, but they do not determine risk-based domain membership.

6 CONCLUSIONS

This study presents a quantitative graph-based method for forming security domains from object-level risk attributes. The method combines continuous risk scoring, discrete security-level mapping, mixed-service conflict resolution, graph partitioning, and boundary insertion. A Python prototype implements the algorithm and separates segmentation decision logic from the enforcement plane.

In the synthetic 150-node case study, the segmented scenario reduced average blast radius from 100% to 18.5% and policy-permitted directed attack paths from 22,350 to 1,240 relative to the flat baseline. These are scenario-specific simulation results. The proposed partition can be enforced through VLAN, Zero Trust, or SDN technologies.

Future work should validate the method on anonymized enterprise flow data, benchmark runtime and memory for larger graphs, export domain assignments to actual enforcement configurations, and analyze sensitivity to risk weights and security-level thresholds.

REFERENCES

[1] IEEE Std 802.1Q-2022, IEEE Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks. IEEE, 2022.

[2] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, Zero Trust Architecture, NIST Special Publication 800-207. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2020, [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207.

[3] Open Networking Foundation, SDN Architecture 1.1, ONF TR-521, Feb. 2016.

[4] FIRST, Common Vulnerability Scoring System Version 4.0: Specification Document, 2023.

[5] J. Zhang, W. Wang, and E. Zio, "Study on the application of graph theory algorithms and attack graphs in cybersecurity assessment," in Proc. 2023 7th International Conference on System Reliability and Safety (ICSRS), Bologna, Italy, pp. 558-564, 2023, [Online]. Available: https://doi.org/10.1109/ICSRS59833.2023.10381005.

[6] L. Bradatsch and F. Kargl, "Integration of security service functions into network-level access control," IEEE Access, vol. 12, pp. 197783-197815, 2024, [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3522575.

[7] S. Sayfullaev, D. Valikulova, and K. Sabirov, "Modern intelligent security analysis systems for corporate networks," in Proc. 2024 5th International Conference on Image Processing and Capsule Networks (ICIPCN), Dhulikhel, Nepal, pp. 717-719, 2024, [Online]. Available: https://doi.org/10.1109/ICIPCN63822.2024.00124.

[8] M. M. Talipov, "Computational modeling and analysis of mechanical power consumption in train assemblers' work," Proceedings of International Conference on Applied Innovation in IT, vol. 13, no. 2, pp. 419-426, 2025, [Online]. Available: https://doi.org/10.25673/120513.

[9] R. Salmorbekova and M. Talipov, "Digital risk matrix and safety management workflow for airport infrastructure in developing countries: a data-driven prioritization approach," Vibroengineering Procedia, vol. 62, pp. 653-661, Jun. 2026, [Online]. Available: https://doi.org/10.21595/vp.2026.26121.

[10] Y. Cao, S. R. Pokhrel, Y. Zhu, et al., "Automation and orchestration of zero trust architecture: Potential solutions and challenges," Machine Intelligence Research, vol. 21, pp. 294-317, 2024, [Online]. Available: https://doi.org/10.1007/s11633-023-1456-2.

[11] H. Moudoud, Z. Abou El Houda, and B. Brik, "Zero trust security architecture for 6G open radio access networks (O-RAN)," IEEE Networking Letters, 2024, [Online]. Available: https://doi.org/10.1109/LNET.2024.3514357.

[12] V. Jain, D. Ather, A. B. Abdul Hamid, R. R. Sharma, G. Talipova, and G. Manteghi, "Secure Arduino-based LiFi communication for IoT sensor networks," in Proc. 2025 Optical Communication, Photonics, Telecommunications, and Intelligent Machine Applications (OPTIMA), Tashkent, Uzbekistan, pp. 189-194, 2025, [Online]. Available: https://doi.org/10.1109/OPTIMA67660.2025.11380401.