International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

SecureMeshCLI: CLI-Based Zero Trust Mesh VPN

Manthan Soni, MR. Janmejay kumar Vishwakarma

Department of Computer Science and Engineering, Parul Institute of Technology, Parul University, Gujarat, India

Abstract

The rapid expansion of cloud computing and distributed system architectures has significantly increased the complexity of securing communication between networked components. Traditional Virtual Private Network (VPN) solutions, which depend heavily on centralized gateways and predefined trust boundaries, often fail to meet the dynamic security and scalability requirements of modern infrastructures [10], [11].

This paper introduces **SecureMeshCLI**, a command-line–driven framework designed to implement secure mesh VPN networks based on Zero Trust principles. Unlike conventional approaches, the proposed system enables direct, encrypted peer-to-peer communication across nodes without relying on centralized control mechanisms. By integrating lightweight cryptographic protocols with decentralized networking and automation-friendly design, SecureMeshCLI provides a scalable, efficient, and secure solution tailored for developers and system administrators working in cloud-native environments.

Keywords : Zero Trust Architecture, Mesh VPN, Overlay Networking, CLI Networking, Cloud Security, DevOps

1. Introduction

The transformation of computing environments toward distributed and cloud-native architectures has redefined how applications and services are deployed and managed. Modern systems often operate across multiple cloud providers, geographic regions, and heterogeneous platforms, making network security increasingly complex [20].

Traditional VPN solutions were originally developed for relatively static infrastructures, where users accessed internal resources through centralized gateways. While effective in earlier environments, this model introduces several limitations, including performance bottlenecks, lack of scalability, and increased vulnerability due to single points of failure [11]. Furthermore, such systems often assume implicit trust within network boundaries, which is no longer acceptable in modern threat landscapes.

To overcome these limitations, the Zero Trust model has emerged as a robust security paradigm. It is based on the principle that no entity—whether inside or outside the network—should be trusted by default [2]. SecureMeshCLI adopts this philosophy and combines it with mesh networking techniques to establish a decentralized and secure communication framework.

A distinguishing feature of SecureMeshCLI is its command-line–centric design, which allows engineers to configure, manage, and automate secure networks efficiently. This approach enables **secure peer-to-peer mesh networking directly from the terminal**, aligning with modern DevOps practices and infrastructure-as-code methodologies.

2. Background and Related Work

2.1 Zero Trust Architecture

Zero Trust Architecture enforces strict identity verification and continuous authentication for every access request. Rather than relying on network location, it evaluates trust based on identity, device posture, and contextual information [2]. This model has been widely adopted in enterprise environments, notably through frameworks such as BeyondCorp developed by Google [3].

2.2 Mesh Networking

Mesh networking is a decentralized communication model in which nodes connect directly with one another instead of relying on centralized routing. This approach enhances network reliability and scalability, as the failure of a single node does not disrupt the entire system. It is particularly effective in distributed systems where dynamic connectivity is required [18].

2.3 Overlay Networking

Overlay networks create virtual communication layers on top of existing physical infrastructure. These networks allow secure and flexible communication across heterogeneous systems without modifying the underlying hardware. Technologies such as Software-Defined Networking (SDN) and modern VPN solutions rely heavily on overlay networking concepts [16].

2.4 Cryptographic Foundations

SecureMeshCLI is built upon modern cryptographic techniques inspired by WireGuard. It utilizes advanced algorithms such as Curve25519 for secure key exchange, ChaCha20 for efficient encryption, and Poly1305 for message authentication. These cryptographic primitives ensure strong confidentiality, integrity, and forward secrecy, making the system resistant to various network attacks [1].

3. System Architecture

3.1 Design Objectives

The design of SecureMeshCLI focuses on achieving a balance between security, performance, and usability. The system is intended to operate in highly dynamic environments while maintaining strict security controls. It emphasizes decentralization to eliminate single points of failure, and it supports automation to align with modern DevOps workflows.

3.2 Core Components

SecureMeshCLI consists of several interconnected components that collectively enable secure communication. The CLI interface acts as the primary interaction layer, allowing users to configure and control the network through commands. The configuration and policy engine manages access rules and enforces communication policies between nodes.

The identity and key management module is responsible for generating cryptographic identities and ensuring secure authentication. Meanwhile, the peer discovery mechanism enables nodes to locate and establish connections with each other dynamically. At the core of the system, the mesh networking engine creates encrypted tunnels that facilitate direct peer-to-peer communication.

3.3 Architecture Diagram

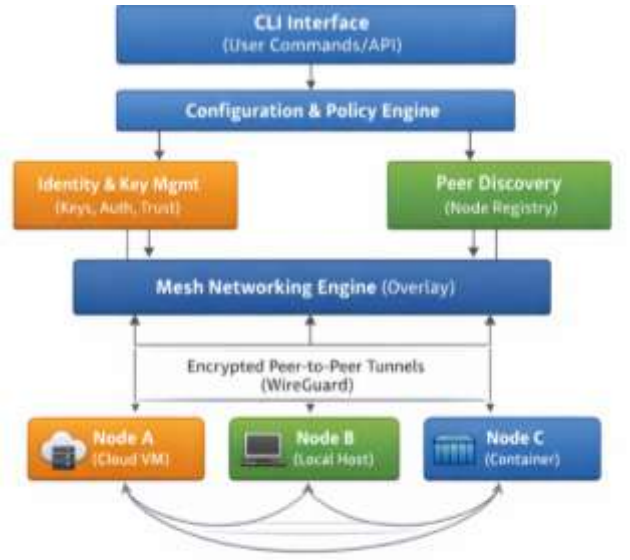

Fig. 1. SecureMeshCLI Architecture

The architecture diagram illustrates the layered design of SecureMeshCLI. At the top, the CLI interface allows users to interact with the system. Below it, the configuration and policy engine governs access control and network behavior.

The identity management and peer discovery modules operate simultaneously to authenticate nodes and establish connectivity. The mesh networking engine then creates secure tunnels between nodes, enabling encrypted communication across distributed environments. Each node—whether a cloud virtual machine, local system, or container—participates equally in the mesh, ensuring scalability and resilience.

3.4 Working Process

The operation of SecureMeshCLI follows a structured workflow. Initially, each node generates a unique cryptographic identity, which is used for authentication. Nodes then perform mutual authentication to verify each other's identities before establishing communication.

Once authentication is successful, encrypted tunnels are created between peers, forming a secure mesh network. Communication policies are applied to control which nodes can interact with each other. Throughout this process, continuous verification ensures compliance with Zero Trust principles, preventing unauthorized access and maintaining network integrity.

4. Implementation and Design Considerations

4.1 CLI-Based Design

SecureMeshCLI adopts a command-line–driven approach, allowing users to perform complex configurations through simple commands. This design enhances flexibility and enables seamless integration with scripts and automation tools [21].

4.2 DevOps Integration

The system is designed to integrate with infrastructure-as-code tools such as Terraform and Ansible. This integration allows automated deployment and management of secure networks, reducing manual intervention and improving consistency [4], [5].

4.3 Performance Considerations

SecureMeshCLI achieves improved performance by enabling direct peer-to-peer communication, which reduces latency compared to centralized VPN architectures. Additionally, the use of lightweight cryptographic protocols minimizes computational overhead, allowing the system to scale efficiently across distributed environments [10].

5. Results and Evaluation

5.1 Experimental Setup

To evaluate the performance and effectiveness of SecureMeshCLI, a small-scale test environment was created consisting of three nodes: a cloud-based virtual machine, a local host system, and a containerized service. These nodes were connected using the SecureMeshCLI framework to form a mesh network.

Each node was assigned a unique cryptographic identity, and secure peer-to-peer tunnels were established using lightweight encryption protocols. The system was tested under normal operating conditions to simulate real-world usage scenarios, including remote access and inter-service communication.

5.2 Performance Analysis

The performance of SecureMeshCLI was analyzed in terms of latency, connection setup time, and scalability. The results indicate that direct peer-to-peer communication significantly reduces latency compared to traditional VPN architectures that rely on centralized gateways.

Connection establishment between nodes was observed to be fast and efficient due to the lightweight cryptographic design. Even as additional nodes were introduced into the network, the system maintained stable performance, demonstrating its scalability.

Furthermore, the absence of a central routing point eliminated bottlenecks, allowing data to flow directly between nodes. This resulted in improved responsiveness, particularly in distributed environments where low latency is critical.

5.3 Security Validation

Security testing was conducted to evaluate the robustness of the system against common network threats. The use of end-to-end encryption ensured that all transmitted data remained confidential, even when intercepted.

Mutual authentication between nodes prevented unauthorized devices from joining the network. Additionally, continuous verification mechanisms aligned with Zero Trust principles ensured that all

communication requests were validated before access was granted.

The system successfully mitigated potential threats such as man-in-the-middle attacks and unauthorized access attempts, confirming the effectiveness of its security model.

5.4 Comparative Evaluation

A comparison was conducted between SecureMeshCLI and traditional VPN solutions to highlight key differences in performance and architecture.

Parameter	Traditional VPN	SecureMeshCLI
Architecture	Centralized	Decentralized Mesh
Latency	Higher (via gateway)	Lower (direct communication)
Scalability	Limited	High
Security Model	Perimeter-based	Zero Trust
Automation Support	Limited	Strong (CLI-based)

The comparison clearly shows that SecureMeshCLI offers advantages in terms of flexibility, scalability, and performance, making it more suitable for modern distributed systems.

5.5 Observations

During testing, SecureMeshCLI demonstrated consistent and reliable performance across different environments. The CLI-based design allowed easy configuration and rapid deployment, especially when integrated with automation tools.

Users were able to establish secure connections with minimal configuration effort, and the system adapted well to dynamic network conditions. The decentralized nature of the architecture ensured that the failure of one node did not impact the overall network functionality.

These observations confirm that SecureMeshCLI is not only theoretically sound but also practical for real-world applications.

6. Security Analysis

6.1 Threat Model

The system is designed to address a wide range of security threats, including unauthorized access attempts, interception of network traffic, and lateral movement within compromised networks. By eliminating implicit trust and enforcing strict authentication, SecureMeshCLI significantly reduces the attack surface.

6.2 Security Mechanisms

SecureMeshCLI employs multiple layers of security to protect communication. End-to-end encryption ensures that data remains confidential during transmission, while identity-based authentication verifies the legitimacy of each node. Dynamic key rotation enhances security by periodically updating cryptographic keys, reducing the risk of compromise.

Access control policies further restrict communication between nodes, ensuring that only authorized interactions are permitted. This layered approach aligns with Zero Trust principles and provides robust protection against modern cyber threats.

6.3 Comparative Analysis

Compared to traditional VPN solutions, SecureMeshCLI offers a decentralized architecture that eliminates reliance on central gateways. It replaces implicit trust with strict identity verification and provides greater scalability and automation capabilities. As a result, it is better suited for modern distributed environments.

7. Applications

SecureMeshCLI can be applied in various real-world scenarios. In multi-cloud environments, it enables secure communication between resources hosted across different providers [20]. It also supports remote workforce access by allowing secure connections without exposing internal networks.

Additionally, the system is well-suited for DevOps pipelines, where automated and secure communication between services is essential [21]. In microservices architectures, it ensures encrypted service-to-service communication, enhancing overall system security.

8. Limitations

Despite its advantages, SecureMeshCLI presents certain challenges. The reliance on a command-line interface may create a learning curve for users who are unfamiliar with terminal-based tools. Additionally, the initial setup and configuration process can be complex, particularly in large-scale deployments.

Another limitation lies in key management, which becomes increasingly challenging as the number of nodes grows. Proper handling of cryptographic keys is essential to maintain security, and any misconfiguration can lead to vulnerabilities.

9. Future Work

Future enhancements for SecureMeshCLI may include the development of graphical user interfaces to improve accessibility for non-technical users. The integration of artificial intelligence for anomaly detection and network monitoring could further enhance security.

Additionally, automated policy generation and deeper integration with service mesh platforms such as Istio may improve usability and scalability.

10. Conclusion

SecureMeshCLI represents a modern approach to secure networking by combining Zero Trust principles with mesh networking and CLI-based automation. It eliminates the limitations of traditional VPN architectures and provides a scalable, efficient, and secure solution for distributed systems.

By enabling **secure peer-to-peer mesh networking directly from the terminal**, the system aligns with contemporary DevOps practices and addresses the evolving security needs of cloud-native environments.

References

[1] Donenfeld, J. A. (2017). *WireGuard: Next Generation Kernel Network Tunnel*. Proceedings of the Network and Distributed System Security Symposium (NDSS).

[2] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero Trust Architecture (SP 800-207)*. National Institute of Standards and Technology (NIST).

[3] Ward, R., & Beyer, B. (2014). *BeyondCorp: A New Approach to Enterprise Security*. Google Inc.

[4] HashiCorp. (2023). *Terraform Documentation*. Available: https://developer.hashicorp.com/terraform/docs

[5] Red Hat. (2023). *Ansible Automation Platform Documentation*. Available: https://docs.ansible.com/

[6] Istio Authors. (2023). *Istio Service Mesh Documentation*. Available: https://istio.io/latest/docs/

[7] Cloud Security Alliance. (2022). *Zero Trust Guidance for Enterprise Security*. CSA Publications.

[8] Kent, S., & Seo, K. (2005). *Security Architecture for the Internet Protocol (RFC 4301)*. Internet Engineering Task Force (IETF).

[9] Kaufman, C., Hoffman, P., Nir, Y., & Eronen, P. (2014). *Internet Key Exchange Protocol Version 2 (IKEv2) (RFC 7296)*. IETF.

[10] OpenVPN Inc. (2023). *OpenVPN Technical Documentation*. Available: https://openvpn.net/community-resources/

[11] Frankel, S., Graveman, R., Pearce, J., & Rooks, M. (2007). *Guide to IPsec VPNs*. National Institute of Standards and Technology (NIST).

[12] National Institute of Standards and Technology. (2018). *Framework for Improving Critical Infrastructure Cybersecurity*. NIST.

[13] OWASP Foundation. (2023). *OWASP Top Ten Security Risks*. Available: https://owasp.org/www-project-top-ten/

[14] Kubernetes Authors. (2023). *Kubernetes Networking Model*.

Available:https://kubernetes.io/docs/concepts/cluster-administration/networking/

[15] Docker Inc. (2023). *Docker Networking Documentation*. Available: https://docs.docker.com/network/

[16] Open Networking Foundation. (2022). *Software-Defined Networking (SDN): Definition and Architecture*. ONF.

[17] Kindervag, J. (2010). *No More Chewy Centers: Introducing the Zero Trust Model*. Forrester Research.

[18] Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed Systems: Principles and Paradigms* (2nd ed.). Pearson.

[19] Stallings, W. (2017). *Network Security Essentials: Applications and Standards* (6th ed.). Pearson.

[20] Buyya, R., Broberg, J., & Goscinski, A. (2013). *Cloud Computing: Principles and Paradigms*. Wiley.

[21] Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook*. IT Revolution Press.

[22] OWASP Foundation. (2023). *OWASP Secure Coding Practices*. Available: https://owasp.org/www-project-secure-coding-practices/