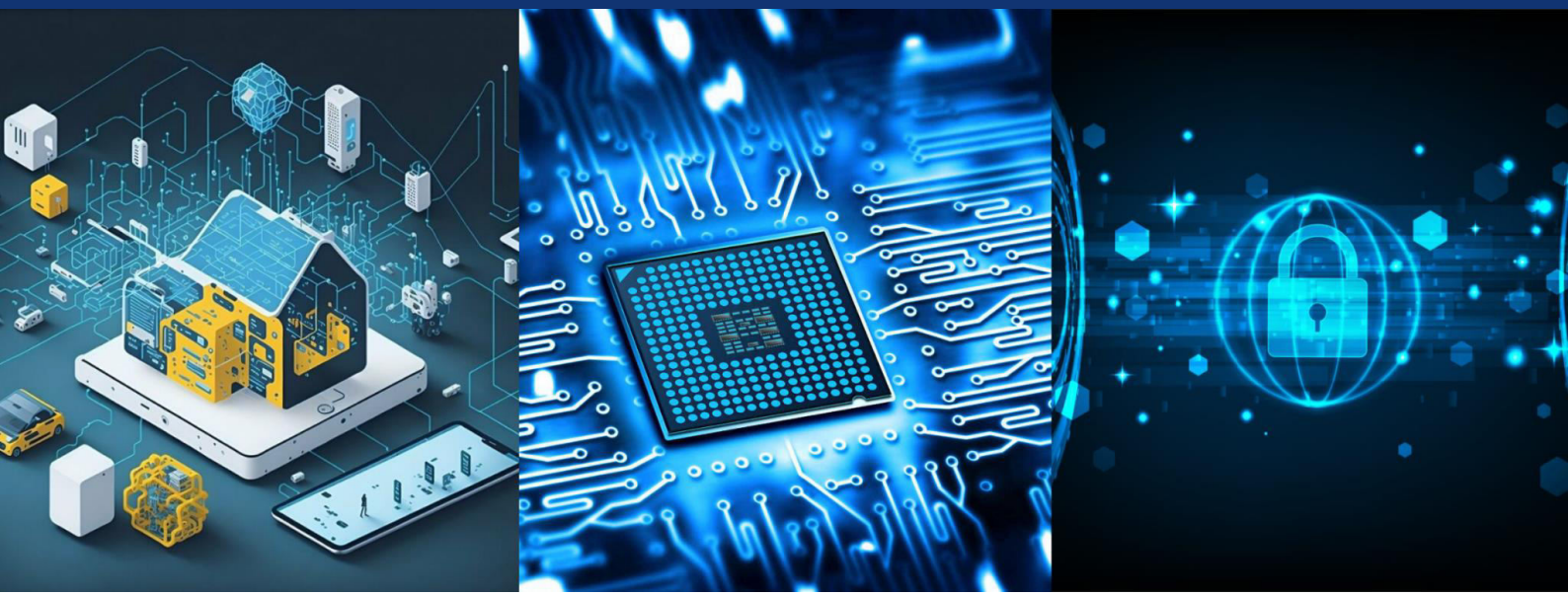
IJIRCCE

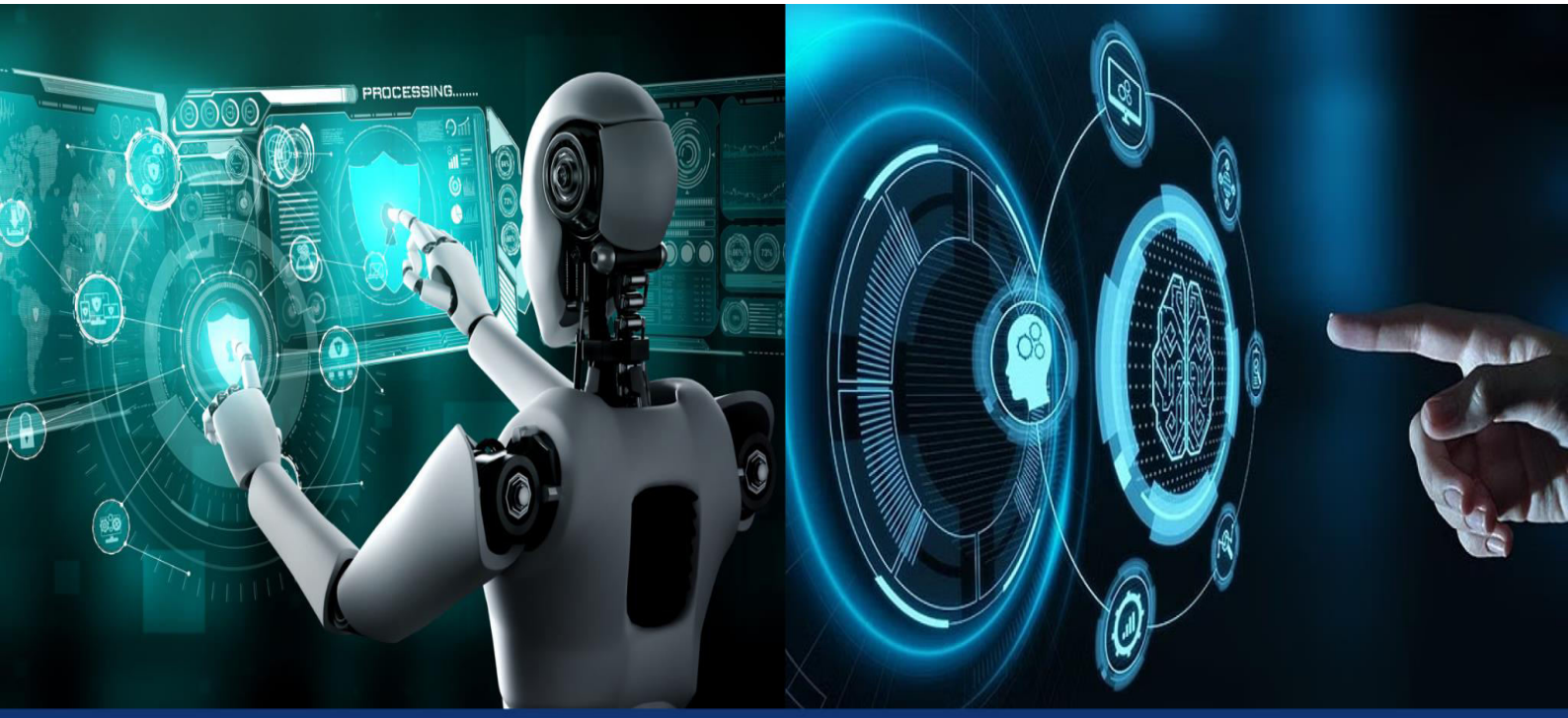
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 7, July 2026

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407054

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

A Zero-Trust Secure Remote Access Framework Using Mutual TLS and Device Posture Verification

Syed Sufyaanuddin, Dr. R. Sridevi

Post-Graduate Student, Department of Computer Science Engineering, Jawaharlal Nehru Technological University, Hyderabad, Telangana, India

Professor, Department of Computer Science Engineering, Jawaharlal Nehru Technological University, Hyderabad, Telangana, India

ABSTRACT: Traditional remote access solutions such as VPNs authenticate users only at login and then grant broad internal network access, exposing organizations to credential theft and lateral movement attacks. This paper proposes a Zero-Trust Secure Remote Access Framework that continuously verifies device identity and security posture before granting access. Mutual TLS (mTLS) restricts connectivity to organization-registered devices through certificate-based authentication, while a Device Posture Agent continuously evaluates antivirus, firewall, operating-system update, and disk-encryption status. The collected posture data is evaluated by a FastAPI-based Policy Engine, which grants application-level access only to compliant devices and redirects non-compliant devices for remediation. Unlike VPNs, which expose the entire internal network, the proposed framework restricts access to individual applications. Experimental results confirm that the framework correctly authenticates registered devices, detects posture violations in real time, and enforces dynamic ALLOW/REMEDIATE/DENY decisions, demonstrating a practical and scalable approach to Zero-Trust remote access.

KEYWORDS: Zero Trust Architecture; Secure Remote Access; Mutual TLS; Device Posture Verification; Certificate-Based Authentication; FastAPI Policy Engine

I. INTRODUCTION

Secure remote access solutions have become increasingly critical as remote work and cloud-based services see widespread enterprise adoption. Traditional Virtual Private Network (VPN) approaches authenticate only the user at the point of login and, once a session is established, typically grant broad access to internal organizational resources without further verifying the security condition of the connecting device. Network Access Control (NAC) systems improve on this only slightly: they perform a posture check during login but do not continuously monitor the device thereafter. Consequently, a device that becomes non-compliant after authentication - for example, through a disabled antivirus program, an outdated operating system, or a deactivated firewall - may continue to hold access privileges for the remainder of the session, creating an avoidable window for credential theft, malware propagation, and lateral movement.

Existing solutions therefore lack continuous compliance verification and rarely reassess security parameters such as antivirus protection, firewall status, patch level, and disk encryption once a session is underway. What is needed is a remote access architecture that continuously verifies both device identity and device security posture before, and throughout, every access session, rather than relying on a single point-in-time authentication event.

This paper proposes a Zero-Trust Secure Remote Access Framework that combines Mutual TLS (mTLS) certificate-based device authentication with continuous Device Posture Verification. A trusted Certificate Authority (CA) issues digital certificates to authorized devices, while a lightweight Device Posture Agent continuously reports antivirus, firewall, operating-system update, and disk-encryption status to a centralized FastAPI-based Policy Engine, which dynamically returns an ALLOW, REMEDIATE, or DENY decision. The specific objectives of this work are to: (i) implement certificate-based device authentication using mTLS; (ii) continuously assess endpoint posture across four security parameters; (iii) develop a FastAPI-based Policy Engine for compliance evaluation and access decisions; (iv)

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407054

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

enforce Zero-Trust principles through continuous rather than one-time verification; and (v) provide a remediation pathway for non-compliant devices, thereby reducing the risk of lateral movement and unauthorized access across enterprise networks.

II. RELATED WORK

Asim et al. [1] proposed SecT, a Zero-Trust Network Access framework for next-generation industrial networks that combines a UDP-based communication architecture with role-based access control and centralized policy management to overcome the latency and scalability limitations of VPNs in Industry 4.0 settings; the work, however, focuses on industrial environments and does not extensively address endpoint posture verification for client devices. Wilcox [2] introduced a hybrid access control model combining dynamic trust evaluation with policy-based authorization and continuous, context-aware verification, but the study concentrates mainly on access control mechanisms with limited discussion of device posture. Bradatsch et al. [3] presented a score-based Zero-Trust access control framework for enterprise networks in which trust scores tied to users and devices are continuously evaluated before network-level access is granted; the approach does not extensively address certificate-based authentication and grows more complex as the network scales. Ma and Chiu [4] proposed a risk-based access control engine for Zero-Trust IoT networks incorporating dynamic trust computation, anomaly detection, and continuous monitoring, though the work targets IoT and sensor environments rather than enterprise-scale remote access.

The principles underlying these works are formalized in the NIST Zero Trust Architecture guidelines [5], which establish that trust should never be granted implicitly and that access must be evaluated per-session based on device and user context. Mutual authentication in the proposed framework is realized using the TLS 1.3 protocol [6], which provides certificate-based mutual verification between communicating endpoints. Collectively, the reviewed works demonstrate strong progress in dynamic, continuous access control but leave a gap in combining lightweight certificate-based device authentication with continuous, multi-parameter endpoint posture verification for general enterprise remote access - the gap this paper addresses.

III. PROPOSED METHODOLOGY

A. Limitations of Existing Systems

Conventional remote access mechanisms exhibit several shortcomings that motivate the proposed design: authentication relies primarily on usernames and passwords, which are susceptible to credential theft and phishing; a successful login grants broad, network-wide access rather than access scoped to a specific application; device posture is, at best, checked once at login and never reassessed for the remainder of the session; there is no mechanism to dynamically restrict access when a device's security state degrades mid-session; and the resulting architecture makes lateral movement straightforward for an attacker who compromises a single authenticated session.

B. Overview of the Proposed Framework

The proposed framework removes this implicit trust by continuously validating both device identity and device security posture before any access to organizational resources is granted. Strong certificate-based authentication is established through Mutual TLS (mTLS): a trusted Certificate Authority issues a unique digital certificate to every authorized device, and the server accepts TLS connections only from clients presenting a valid certificate. Independently of this transport-layer handshake, a Device Posture Agent installed on the endpoint continuously monitors four security parameters - antivirus status, firewall status, operating-system update status, and disk-encryption status - and periodically reports them, together with the device certificate, to a centralized Policy Engine. Access is granted only when both checks succeed: the device must hold a valid certificate and must currently satisfy the organization's compliance policy.

C. System Architecture

Fig. 1 shows the architecture of the proposed framework. The Certificate Authority (CA) is the framework's trust anchor: it generates and issues digital certificates that establish each device's identity and underpin the mTLS handshake. The Device Posture Agent, installed on each endpoint, continuously collects the four posture parameters described above and packages them, together with the device's certificate, into a posture report transmitted to the server over HTTPS. The Verification Module validates the presented certificate and confirms device identity before the request is allowed to proceed, preventing unregistered devices from participating in the access process. The FastAPI Policy Engine then evaluates the reported posture against the organization's compliance policy and generates one of three access decisions -

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407054

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

ALLOW, REMEDIATE, or DENY. Compliant devices are granted access to the requested Application Resource, while non-compliant devices are redirected to a Remediation Interface that guides the user through corrective steps such as enabling antivirus protection, activating the firewall, installing pending updates, or enabling disk encryption. Because posture is re-evaluated on every request rather than once at login, a device that drifts out of compliance mid-session loses access on its very next check, sharply reducing the window available for lateral movement.

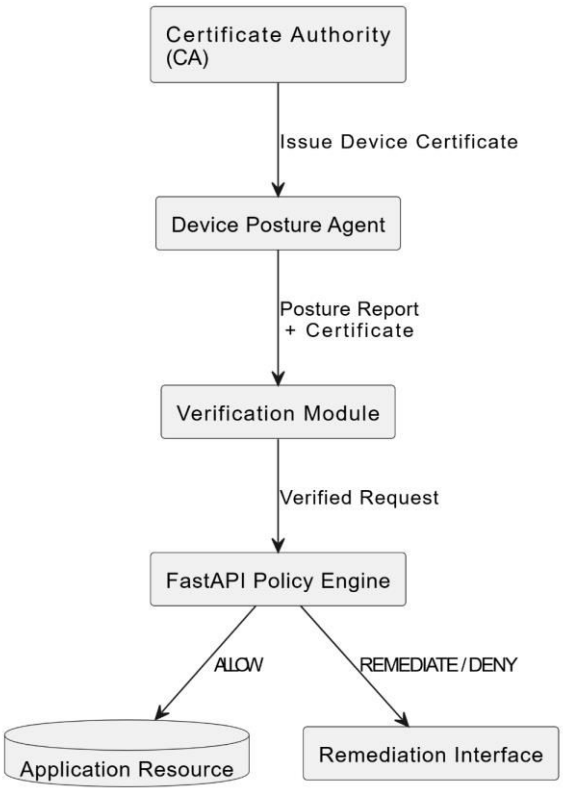


Fig. 1. System architecture of the proposed Zero-Trust secure remote access framework.

The interaction among framework components is illustrated by the sequence diagram in Fig. 2. A client device first obtains a signed certificate from the CA and uses its Device Agent to collect posture information; the device certificate and posture report are then sent to the FastAPI server, which forwards the verified request to the Policy Engine. The Policy Engine verifies the certificate and device identity, evaluates the reported posture against the compliance policy, and returns a decision. When the decision is ALLOW, the server requests access to the target application resource on the device's behalf and relays the resulting access grant back to the client; otherwise, the device is directed to remediation. This request-response cycle repeats at every monitoring interval, giving the framework continuous rather than one-time verification.

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407054

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

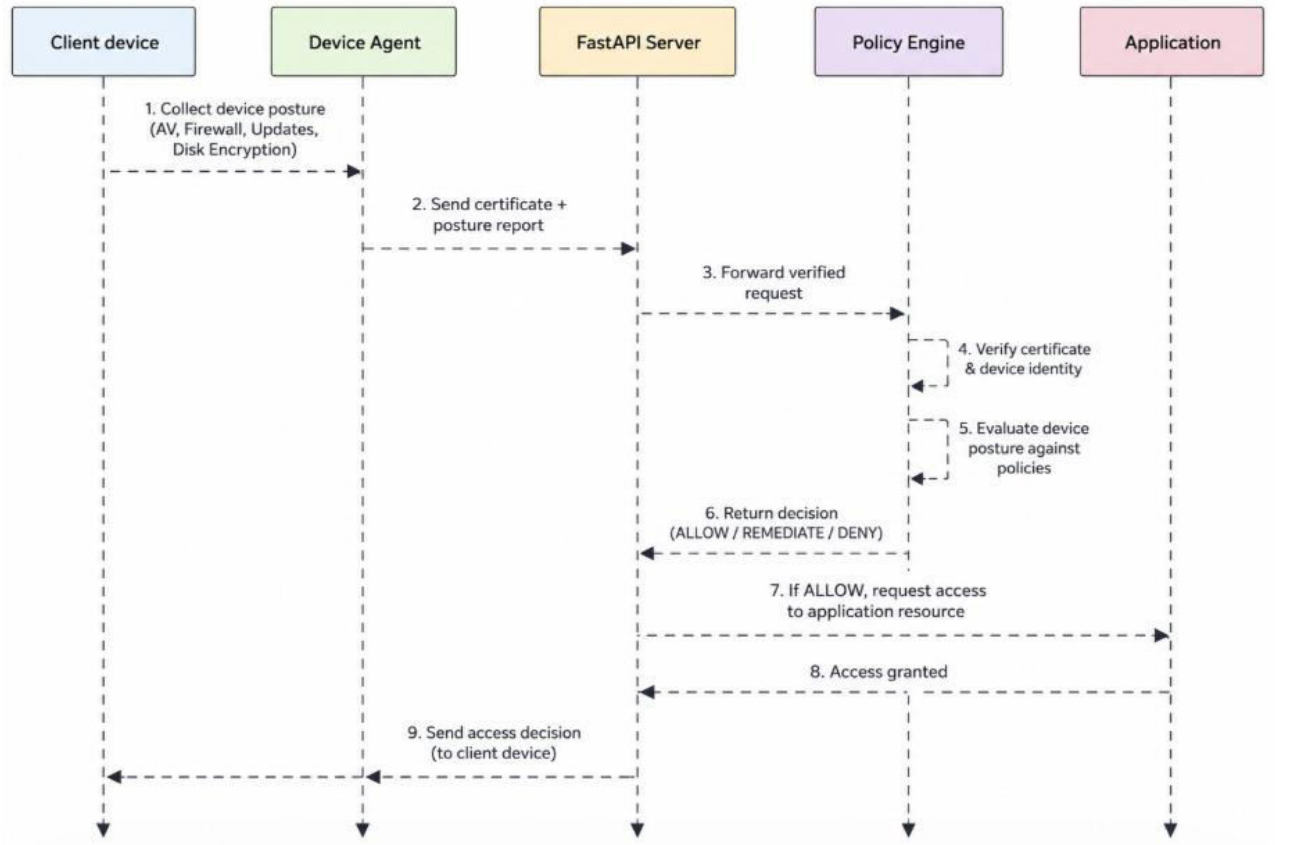


Fig. 2. Sequence diagram of the device authentication and access-decision process.

D. Policy Decision Algorithm

Algorithm 1 summarizes the decision logic implemented by the Policy Engine. A device is denied at the transport layer if it cannot complete the mTLS handshake with a certificate issued by the trusted CA; otherwise, its most recently reported posture is evaluated against the compliance policy.

Algorithm 1: Access Decision Evaluation

Input: Device certificate **C**, posture report **P**

Output: Decision ∈ {ALLOW, REMEDIATE, DENY}

```
if mTLS authentication fails or C is not trusted then
   return DENY
end if

if P.antivirus ∧ P.firewall ∧
  P.os_updated ∧ P.disk_encryption then
   return ALLOW
else
   return REMEDIATE
end if
```

E. Implementation

The framework was implemented and evaluated on Windows 10/11 endpoints using Python 3.x throughout. The FastAPI Policy Engine and its REST endpoints (/access and /devices) were hosted using the Uvicorn ASGI server over HTTPS, with server and CA certificates generated and managed through OpenSSL. The Device Posture Agent uses PowerShell commands - (Get-MpComputerStatus).RealTimeProtectionEnabled, Get-NetFirewallProfile, manage-bde -status, and Get-HotFix - to sample antivirus, firewall, disk-encryption, and patch status respectively, and transmits the resulting

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407054

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

JSON posture report to the server every five seconds using the Python Requests library over HTTPS. For centrally managed endpoints, a companion collector module performs subnet-wide device discovery and posture collection over Windows Remote Management (WinRM), invoking the same PowerShell checks remotely through Invoke-Command so that devices without a locally installed agent can still be assessed. FastAPI's automatically generated Swagger UI was used throughout development to exercise the /access and /devices endpoints, inspect posture reports, and observe access decisions in real time.

IV. EXPERIMENTAL RESULTS

A. Certificate Authority and mTLS Setup

A self-signed root certificate was first generated for the Certificate Authority using openssl genrsa followed by openssl req -x509, after which server and device certificates were issued and signed by this CA. Inspecting the generated CA certificate (openssl x509 -text -noout) confirmed a 2048-bit RSA key, SHA-256 signature algorithm, and a 365-day validity period. The FastAPI server was then launched via Uvicorn using the issued server key and certificate (--ssl-keyfile, --ssl-certfile), so that only clients presenting a certificate signed by the trusted CA could complete the TLS handshake.

B. Device Posture Verification and Policy Evaluation

Fig. 3 shows representative output from the Device Posture Agent and Policy Engine during continuous monitoring. When all four posture parameters were satisfied, the Policy Engine consistently returned an ALLOW decision (HTTP 200, {"decision":"ALLOW"}); when the firewall was deliberately disabled on the test endpoint, the very next posture report - transmitted within the five-second monitoring interval - was correctly evaluated as REMEDIATE, confirming that the framework detects posture drift and revokes effective access without waiting for re-authentication.

```
F:\new\agent>python agent.py
UPDATE RAW: '10 June 2026 00:00:00'

Change Detected: {'device_id': 'Sufyaan', 'antivirus': True, 'firewall': True, 'os_updated': True, 'disk_encryption': True, 'timestamp': '2026-06-18T09:56:36.121889'}
STATUS: 200
TEXT: {"decision":"ALLOW"}
Server Decision: ALLOW
UPDATE RAW: '10 June 2026 00:00:00'

Change Detected: {'device_id': 'Sufyaan', 'antivirus': True, 'firewall': True, 'os_updated': True, 'disk_encryption': True, 'timestamp': '2026-06-18T09:56:47.316007'}
STATUS: 200
TEXT: {"decision":"ALLOW"}
Server Decision: ALLOW
UPDATE RAW: '10 June 2026 00:00:00'

Change Detected: {'device_id': 'Sufyaan', 'antivirus': True, 'firewall': True, 'os_updated': True, 'disk_encryption': True, 'timestamp': '2026-06-18T09:56:58.645356'}
STATUS: 200
TEXT: {"decision":"ALLOW"}
Server Decision: ALLOW
UPDATE RAW: '10 June 2026 00:00:00'

Change Detected: {'device_id': 'Sufyaan', 'antivirus': True, 'firewall': True, 'os_updated': True, 'disk_encryption': True, 'timestamp': '2026-06-18T09:57:10.081329'}
STATUS: 200
TEXT: {"decision":"ALLOW"}
Server Decision: ALLOW
UPDATE RAW: '10 June 2026 00:00:00'

Change Detected: {'device_id': 'Sufyaan', 'antivirus': True, 'firewall': False, 'os_updated': True, 'disk_encryption': True, 'timestamp': '2026-06-18T09:57:21.359572'}
STATUS: 200
TEXT: {"decision":"REMEDIATE"}
Server Decision: REMEDIATE
UPDATE RAW: '10 June 2026 00:00:00'
```

Fig. 3. Sample Policy Engine output showing ALLOW and REMEDIATE decisions during continuous monitoring.

C. Device Discovery and Continuous Monitoring

The collector module was validated by scanning a local /24 subnet over WinRM; reachable hosts were discovered automatically via Test-WSMan, and their antivirus, firewall, disk-encryption, and update status were retrieved remotely through Invoke-Command and forwarded to the Policy Engine, which returned ALLOW and REMEDIATE decisions consistent with each host's actual configuration. This confirms that the framework can assess endpoints centrally without requiring a locally installed agent on every device.

D. API and Interface Testing

The FastAPI Swagger UI was used to exercise the /access endpoint directly: posting a compliant posture payload returned HTTP 200 with {"decision":"ALLOW"}, while the /devices endpoint correctly aggregated and returned the latest posture record and decision for every device that had reported to the server, confirming that the administrator-facing monitoring interface accurately reflects the current compliance state of the environment.

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407054

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

E. Summary of Testing

Table I summarizes the outcomes of the four testing phases carried out on the framework.

TABLE I. SUMMARY OF TESTING RESULTS

Testing Type	Key Result
Unit	Each posture check and the certificate validator correctly distinguished compliant/valid from non-compliant/invalid cases.
Integration	Agent-server, mTLS + policy evaluation, and collector-server discovery paths all worked without errors.
System	Correct ALLOW/REMEDIATE decisions were generated across all tested non-compliance scenarios; posture changes were reflected in real time.
Performance	Near real-time decision latency; discovery and evaluation remained stable under increased/concurrent load.

V. CONCLUSION AND FUTURE WORK

This paper presented a Zero-Trust Secure Remote Access Framework that pairs certificate-based Mutual TLS authentication with continuous, multi-parameter device posture verification, replacing the one-time, network-wide trust granted by conventional VPNs with per-session, application-scoped access decisions. The FastAPI-based Policy Engine was shown to correctly enforce ALLOW and REMEDIATE decisions based on real-time antivirus, firewall, operating-system update, and disk-encryption status, while unauthorized devices lacking a valid certificate are denied at the transport layer. Experimental evaluation confirmed that the framework detects posture drift within a single monitoring interval, supports centralized device discovery and assessment over WinRM, and processes access decisions with minimal latency, demonstrating a practical and scalable approach to Zero-Trust remote access.

Future work will focus on three directions: incorporating dynamic, risk-based trust scoring that adapts to user behavior, device activity, location, and threat-intelligence feeds rather than static compliance rules; extending posture collection beyond Windows to Linux, macOS, mobile, and IoT endpoints to broaden deployment across heterogeneous enterprise environments; and integrating the Policy Engine with Security Information and Event Management (SIEM) platforms to enable centralized logging, security analytics, and automated incident response.

REFERENCES

[1] M. Asim, N. Tariq, A. I. Awad, F. Waheed, U. Ullah, and G. Murtaza, “SecT: A Zero-Trust Framework for Secure Remote Access in Next-Generation Industrial Networks,” IEEE Journal on Selected Areas in Communications, vol. 43, no. 6, pp. 2293–2311, 2025.
[2] C. Wilcox, “Towards a Zero Trust Based Hybrid Access Control Model,” IEEE, 2024.
[3] L. Bradatsch et al., “Zero Trust Score-Based Network-Level Access Control in Enterprise Networks,” IEEE, 2023.
[4] Y. W. Ma and P. H. Chiu, “A Novel Risk-Based Access Control Engine in Zero Trust Architecture for IoT Networks,” International Journal of Information Security, 2025.
[5] N. Mavroeidis and S. Bromander, “Cyber Threat Intelligence Model for Network Security,” IEEE Access, vol. 5, pp. 12653–12671, 2017.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details