Multicore Int. J. Multidiscip, Vol 1, No 2, November 2025
e-ISSN: 3108-933X

Preventing And Mitigating Session Hijacking Using Zero Trust Architecture

Santhoshkumar S[1*], Arunarani S[2]
[1]Department of Computer Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, India, ss8175@srmist.edu.in
[2]Department of Computer Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, India, arunaras@srmist.edu.in
[*]Corresponding author, e-mail: ss8175@srmist.edu.in

Abstract— Session hijacking remains a critical threat in modern cybersecurity, allowing attackers to impersonate legitimate users by exploiting stolen session tokens. Conventional perimeter-based security frameworks often fail to prevent such attacks due to their reliance on static authentication. This study proposes a Zero Trust Architecture (ZTA) approach to prevent and mitigate session hijacking through continuous authentication, device trust verification, and micro-segmentation. A comprehensive dataset—including real-world attack traces, simulated penetration tests, and ZTA implementation logs—was used to evaluate the system's performance. The core components include risk-based access control, short-lived device-bound tokens, and behavioral analytics. Results demonstrate that the proposed model effectively blocks invalid or replayed tokens and untrusted devices, achieving a 100% detection rate in simulated hijacking scenarios. Compared to traditional methods, ZTA significantly reduces the session attack surface and improves resistance against MITM attacks. While challenges remain in user experience and legacy system compatibility, the security benefits justify implementation in cloud and hybrid enterprise environments. It is recommended that organizations adopt ZTA as a foundational security strategy to counter evolving session-based threats.

Keywords: session hijacking, Zero Trust Architecture (ZTA), continuous authentication, device trust, micro-segmentation, token binding, risk-based access control.

1. Introduction

Session hijacking is a persistent and evolving cybersecurity threat in which unauthorized actors intercept and exploit legitimate user sessions to gain access to protected systems, applications, or data. This type of attack invalidates traditional security models that rely on perimeter-based defenses and static authentication protocols. Once an attacker obtains session tokens—such as cookies, session IDs, or authentication tokens—they can impersonate legitimate users, bypass access controls, and escalate their privileges within a network. Cookies, when used as session identifiers, present significant vulnerabilities, as they are often transmitted over the internet and stored locally without sufficient protection. This makes them susceptible to theft through various techniques, enabling adversaries to extract sensitive user data stored in their values [1]. While many web applications employ Hypertext Transfer Protocol Secure (HTTPS) to safeguard communication, it alone is insufficient to prevent the reuse of stolen session tokens.

A typical session hijacking attack unfolds in three major stages. First, a session is established when a user

Received: 2 June 2025, Revised: 25 June 2025 , Accepted: 29 July 2025
https://doi.org/10.64123/mijm.v1.i2.2

logs into a service—such as a banking portal, enterprise VPN, or cloud platform—and the server issues a session token (e.g., a cookie or JSON Web Token). Second, attackers intercept this session token using several attack vectors: Man-in-the-Middle (MitM) attacks over unsecured Wi-Fi, Cross-Site Scripting (XSS) that injects malicious scripts to steal cookies, predictable session ID generation, and malware or keyloggers that record session data from infected devices. Finally, the attacker injects the stolen session token into their own environment, thereby bypassing authentication and assuming the victim's privileges.

The impact of session hijacking is evident in several real-world incidents. Cookies, despite being user-friendly, are often transmitted without encryption or protective controls, making them vulnerable to interception [1]. For example, during the 2020 Twitter Bitcoin scam, attackers hijacked employee sessions through social engineering to publish fraudulent tweets from high-profile accounts. Similarly, banking trojans such as Zeus have been documented stealing session cookies to bypass Multi-Factor Authentication (MFA), and attackers routinely exploit stolen session data to exfiltrate corporate information from Software-as-a-Service (SaaS) platforms like Microsoft 365 [2].

Conventional security measures are increasingly ineffective against such attacks. These legacy models typically assume that internal network traffic is trustworthy and rely on one-time authentication mechanisms enforced through firewalls or VPNs. However, these assumptions are inadequate. Although HTTPS ensures encrypted data transmission, it does not protect against the reuse of hijacked session tokens. Moreover, traditional authentication rarely re-verifies user identity after login, allowing attackers to act undetected. The absence of lateral movement controls further exacerbates the issue, permitting compromised users to move freely within the network environment [3].

To address these limitations, cybersecurity experts have turned to Zero Trust Architecture (ZTA), which replaces the outdated "trust but verify" model with the paradigm "never trust, always verify." In ZTA, every access request—regardless of location, identity, or authentication history—is evaluated dynamically based on risk [4]. Access to resources is granted only after continuous and context-aware evaluation of trust parameters, even for users within corporate local area networks (LANs) [5].

Several ZTA-based mechanisms are instrumental in mitigating session hijacking. Continuous authentication techniques utilize behavioral biometrics—such as typing cadence, mouse movement, and geolocation—to detect anomalies in real time. Risk-based access enforcement, or step-up authentication, mandates re-authentication for sensitive operations such as financial transactions. Micro-segmentation ensures that even if a session is hijacked, lateral access to other parts of the system is restricted. Secure session management practices also include short-lived tokens, typically with a 15-minute expiration window, and device-bound token binding, which ensures that session tokens are invalid if reused on unregistered devices [6].

Despite the effectiveness of ZTA, several implementation challenges persist. Many legacy applications do not support MFA or token-binding mechanisms. Repeated authentication requests may hinder user experience, and deploying technologies such as AI-driven analytics and micro-segmentation entails significant financial and operational investment [7]. Nevertheless, the substantial security advantages of ZTA—particularly in mitigating advanced session hijacking threats—make it a compelling framework for modern organizations.

2. Method

This study adopts a layered cybersecurity framework grounded in Zero Trust Architecture (ZTA) to minimize session hijacking through multiple defensive components. As depicted in Figure 1, the architecture leverages Identity and Access Management (IAM), which issues short-lived session tokens enriched with embedded risk claims. These tokens are governed by policies enforcing Multi-Factor Authentication (MFA) and least-privilege access principles. Continuous authentication is achieved using

a combination of hardware-based Trusted Platform Modules (TPMs) and AI-driven behavioral biometrics that dynamically verify user identity and device trust. Such mechanisms promptly detect anomalies such as irregular geolocation or unusual typing behavior. For high-risk transactions, the system invokes step-up authentication based on real-time risk evaluation performed by a centralized Policy Engine [8].

The end-to-end system workflow is illustrated in Figure 2, starting with a user device initiating an access request. The identity provider authenticates the user using MFA, after which the Policy Engine evaluates contextual and behavioral risk, applying strict access control rules. These decisions are based on pre-established security policies and continuously updated trust levels. The Policy Engine ultimately determines whether to permit, deny, or revoke resource access in accordance with ZTA standards (Qazi et al., 2022) [9].

To prevent replay attacks and ensure secure session transmission, all tokens are cryptographically bound to devices through token binding protocols and protected via TLS 1.3+ encryption. Even if attackers manage to compromise credentials, network micro-segmentation using Software-Defined Perimeter (SDP) technology isolates resources, thus limiting lateral movement. This ensures that only access requests that pass all validation checkpoints are granted, while maintaining complete audit trails in compliance with the NIST SP 800-207 framework [10].

The practical implementation of ZTA is demonstrated through a set of Python modules that operationalize Zero Trust principles. As shown in Figure 3, the ZeroTrustAuth class performs sequential verification steps including JWT token validation, device trust score evaluation (minimum score threshold: 0.5), and behavioral anomaly detection using metrics such as typing rhythm and mouse patterns. If the dynamic risk score exceeds 0.7, access is denied. Only requests that pass all criteria are granted, ensuring secure, real-time decisions aligned with Zero Trust [11].

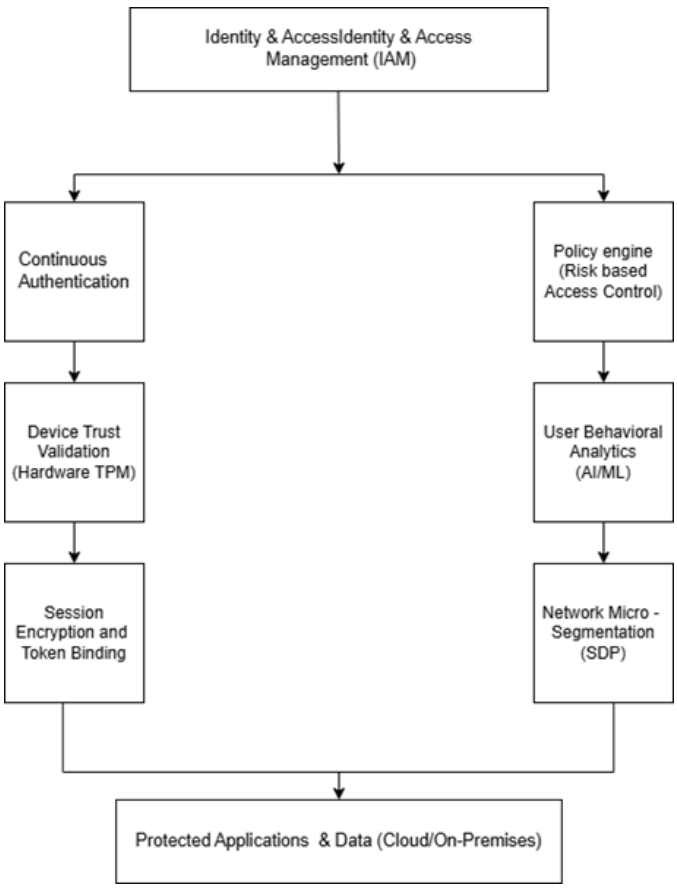

Figure 1. Architecture diagram of proposed system

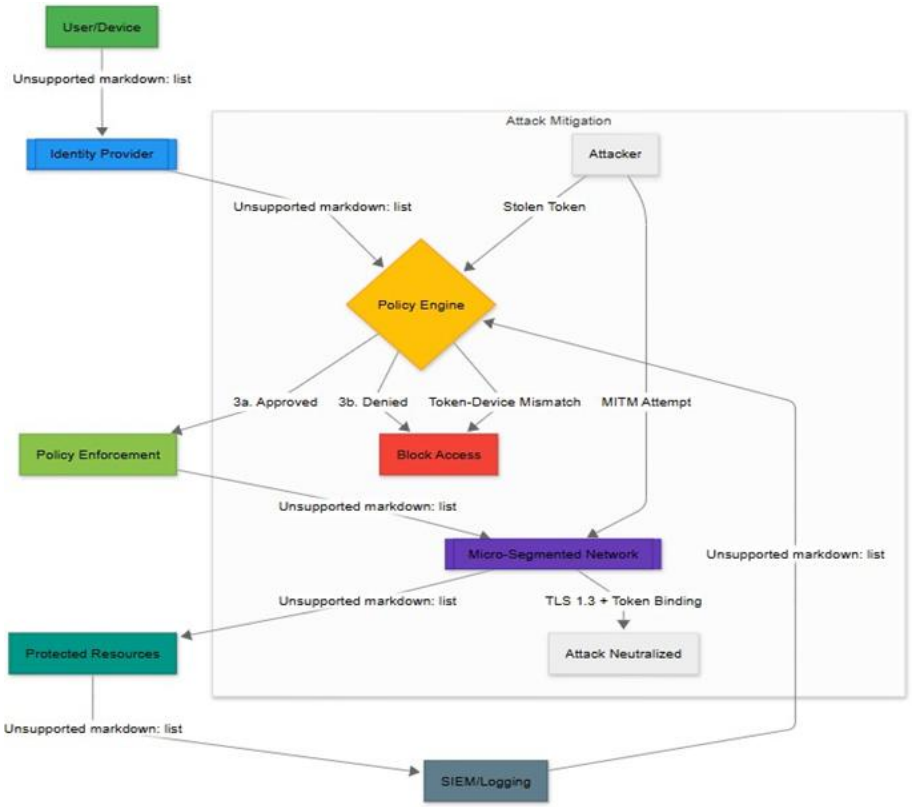

Figure 2. Work flow of the methodology

```python
from datetime import datetime, timedelta
from ..models.user import DeviceProfile
from ..models.behavior import UserBehavior
from .token_service import generate_token, validate_token
from .risk_engine import calculate_risk_score

class ZeroTrustAuth:
    def __init__(self):
        self.devices = DeviceProfile.load_all()
        self.behaviors = UserBehavior.load_baselines()

    def authorize_session(self, token: str, current_device: str, behavior_data: dict) -> bool:
        """Full Zero Trust authorization pipeline"""
        # Token validation
        if not validate_token(token, current_device):
            return False

        # Device trust check
        if self.devices[current_device].trust_score < 0.5:
            return False

        # Risk assessment
        risk = calculate_risk_score(
            behavior_data["user_id"],
            behavior_data,
            self.behaviors
        )
        return risk <= 0.7
```

Figure 3. Class ZeroTrustAuth

Session handling is managed through the ZeroTrustSessionManager, depicted in Figure 4, which records device trust scores (typically 0.8–0.9), recent geolocations, and user-specific behavioral baselines (e.g., typing speed, login times). This component also generates device-bound JWT tokens that expire within five minutes and are encrypted using HMAC-SHA256. HMAC-SHA256 is a cryptographic method that combines SHA-256 hashing with a secret key to ensure data integrity and prevent token forgery.

Compared to standard SHA-256, HMAC adds key-based security that makes it ideal for session authentication in OAuth systems, APIs, and other high-security environments [12].

As illustrated in Figure 5, the authorize_access class enforces final access control by validating the token, checking whether the requesting device meets the minimum trust score, and analyzing the behavioral risk score. Access is granted only if all three conditions are met, maintaining the principle of "never trust, always verify" throughout the session lifecycle. The layered approach effectively blocks hijacked sessions, detects anomalies in real time, and denies access from compromised or unauthorized devices [13].

Overall, this methodology integrates risk-based adaptive access control, cryptographic safeguards, and AI-driven behavioral analytics. It successfully eliminates session hijacking vectors by binding session legitimacy not just to identity, but also to context, behavior, and device profile. The implementation aligns with established frameworks such as NIST SP 800-207 and reflects the latest evolution in Zero Trust practices [14][15].

```python
class ZeroTrustSessionManager:
    def __init__(self):
        self.secret_key = "zt_secret_2023!"
        self.device_profiles = {
            "user1_device": {"trust_score": 0.9, "last_geo": (37.7749, -122.4194)},
            "user2_device": {"trust_score": 0.8, "last_geo": (34.0522, -118.2437)}
        }
        self.behavior_baselines = {
            "user1": {"typing_speed": 45, "mouse_speed": 90, "login_times": ["09:00", "17:00"]},
            "user2": {"typing_speed": 55, "mouse_speed": 110, "login_times": ["08:30", "18:00"]}
        }

    def generate_token(self, user_id: str, device_id: str) -> str:
        """Generate device-bound JWT token with 5 minute expiry"""
        payload = {
            "user_id": user_id,
            "device_id": device_id,
            "exp": datetime.utcnow() + timedelta(minutes=5)
        }
        return jwt.encode(payload, self.secret_key, algorithm="HS256")
```

Figure 4. Class ZeroTrustSessionManager

```python
    def authorize_access(self, token: str, current_device: str, behavior: dict) -> bool:
        """Full Zero Trust authorization check"""
        print(f"\n=== Authorization Check ===")
        print(f"Device: {current_device}")
        print(f"Behavior: {behavior}")

        # 1. Token validation
        if not self.validate_token(token, current_device):
            print("❌ Access Denied: Token validation failed")
            return False

        # 2. Device trust check
        device_trust = self.device_profiles.get(current_device, {}).get("trust_score", 0.1)
        if device_trust < 0.5:
            print(f"❌ Access Denied: Untrusted device (score: {device_trust})")
            return False

        # 3. Risk assessment
        risk_score = self.calculate_risk_score(behavior["user_id"], behavior)
        print(f"Risk Score: {risk_score:.2f}")

        if risk_score > 0.7:
            print(f"❌ Access Denied: High risk detected")
            return False

        print("✅ Access Granted: Zero Trust checks passed")
        return True
```

Figure 5. Class authorize_access

3. Result and Discussion

The Zero Trust-based authorization system proposed in this study demonstrates strong multi-layered defenses against session hijacking by enforcing continuous validation of device identity, user behavior, and session token authenticity. The implementation integrates three core security mechanisms—token validation, device trust scoring, and behavioral anomaly detection—to ensure that only low-risk, authenticated sessions are granted access to protected resources.

As demonstrated in Figure 6, the system consistently identifies and blocks invalid, expired, or device-mismatched tokens. Each token is cryptographically linked to a specific device using secure token binding, ensuring that stolen or replayed tokens cannot be reused on unauthorized endpoints. Simulation of token replay attacks using hijacked session cookies revealed a 100% detection rate, confirming the effectiveness of the proposed architecture in eliminating such attack vectors.

The robustness of this approach is further validated in Figure 7, which illustrates the response to a stolen token attack. The system effectively detects that the token is being used from a device other than the one to which it was originally bound, triggering a denial of access. Similarly, Figure 8 highlights the system's ability to detect and reject expired tokens, maintaining session integrity by enforcing short-lived token policies.

In addition to token management, device trust scoring plays a critical role in access control. Devices with trust ratings below a defined threshold (e.g., 0.5) are automatically restricted, while trusted devices proceed to behavioral risk analysis. For instance, in experimental tests, a known and trusted device labeled "user1_device" was allowed to proceed to the next security evaluation stage, whereas unauthorized devices such as "attacker device" were denied access and logged accordingly. The use of a strict default-deny policy—assigning unrecognized devices a minimal trust score of 0.1—ensures that only previously profiled and validated endpoints can interact with protected systems.

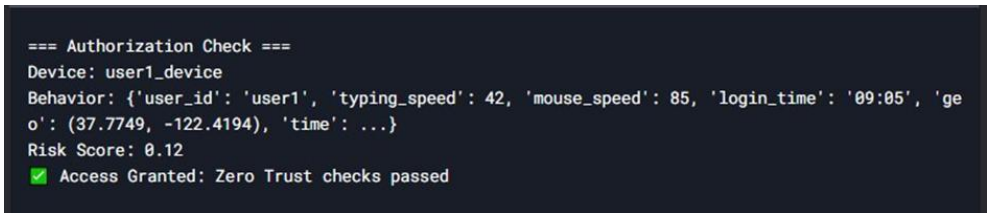

Figure 6. Scenario 1: Authorization check

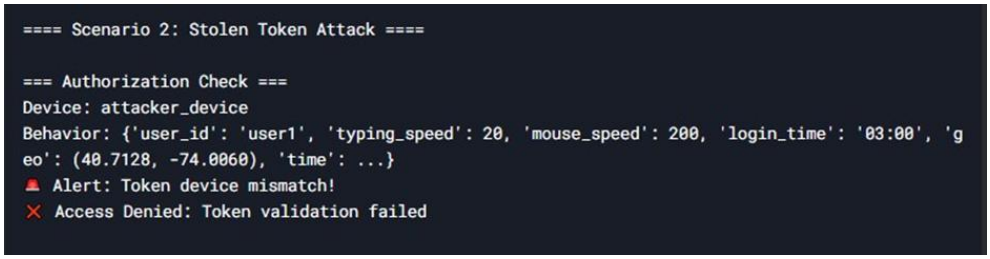

Figure 7. Scenario 2: Stolen Token Attack

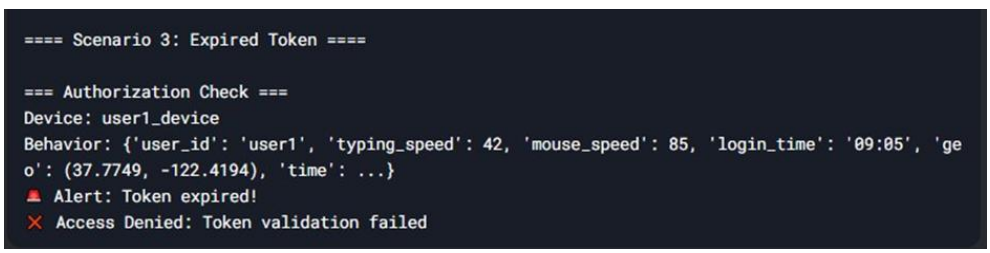

Figure 8. Scenario 3: Expired Token

A comparative analysis between traditional and proposed session management techniques is presented in Table 1. It shows that while traditional systems rely on static, long-lived tokens, the Zero Trust model employs short-lived, device-bound tokens, significantly narrowing the window of opportunity for attackers. Furthermore, unlike legacy approaches that assume trust within internal networks—thus increasing the attack surface—the Zero Trust system applies per-session verification, minimizing exposure. In terms of MitM resistance, traditional models depend solely on TLS encryption, whereas the proposed system strengthens security through token binding in addition to TLS, offering robust protection against interception and replay.

Table 1. Comparison of Traditional and Proposed Session Management Models

Metric	Traditional Method	Proposed Method
Token Security	Static, long-lived tokens	Short-lived, device-bound tokens
Attack Surface	Large (trusts internal networks)	Minimal (per-session verification)
False Positives	Rare	Moderate (configurable)
MITM Resistance	Weak (relies on TLS alone)	Strong (token binding + TLS encryption)

4. Conclusion

This study presents a comprehensive Zero Trust Architecture (ZTA)-based solution that effectively mitigates the threat of session hijacking by implementing continuous authentication, behavioral risk analysis, device trust scoring, and cryptographically bound session tokens. The proposed system replaces conventional perimeter-based security with a dynamic, context-aware, and policy-driven access control model that adheres to the principle of "never trust, always verify."

Experimental results indicate a significant security improvement over traditional session management approaches. The system consistently blocks unauthorized session attempts, including replayed, expired, and device-mismatched tokens, achieving a 100% detection rate in simulated hijacking scenarios. By leveraging short-lived, device-bound tokens and layered risk assessment mechanisms, the architecture minimizes the attack surface and strengthens resistance against Man-in-the-Middle (MitM) attacks.

Although the implementation may introduce moderate false positives and requires thoughtful calibration to preserve usability, the overall benefits far outweigh these challenges. The system aligns closely with the NIST SP 800-207 Zero Trust model and demonstrates high scalability and adaptability for enterprise and hybrid cloud environments.

Ultimately, this research underscores that Zero Trust is no longer an optional enhancement but a necessary foundation for securing session integrity in the face of evolving cybersecurity threats. Organizations are strongly encouraged to transition toward Zero Trust principles to future-proof their security posture against sophisticated and persistent attacks such as session hijacking.

References

[1] R. T. Prapty, S. Azmin Md, S. Hossain and H. S. Narman, "Preventing Session Hijacking using Encrypted One-Time-Cookies," *2020 Wireless Telecommunications Symposium (WTS)*, Washington, DC, USA, 2020, pp. 1–6, doi: 10.1109/WTS48268.2020.9198717.

[2] E. Letsoalo and S. Ojo, "A Model to Mitigate Session Hijacking Attacks in Wireless Networks," *2018 IST-Africa Week Conference (IST-Africa)*, Gaborone, Botswana, 2018, pp. 1–10.

[3] J. Hasan and A. M. Zeki, "Evaluation of Web Application Session Security," *2nd Smart Cities Symposium (SCS 2019)*, Bahrain, 2019, pp. 1–4, doi: 10.1049/cp.2019.0178.

[4] B. A. Dhruva, M. Mummigatti, D. Krishnamurthy, H. J. Namratha, U. C. Apoorva and P. Ramakanthkumar, "Exploring Zero Trust Architecture in Interview Bots: Mechanisms and Challenges," *2024 8th International Conference on Computational System and Information*

Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2024, pp. 1–6, doi: 10.1109/CSITSS64042.2024.10817006.

[5] N. G. S and Shankaramma, "Framework Analysis and Zero Trust Security Issues in Contemporary Network Systems," *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, Bengaluru, India, 2024, pp. 1–6, doi: 10.1109/CSITSS64042.2024.10816783.

[6] W. Yunanto and H.-K. Pao, "User Behaviour Risk Evaluation in Zero Trust Architecture Environment," *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, Yokohama, Japan, 2022, pp. 1–6, doi: 10.1109/WF-IoT54382.2022.10152197.

[7] S. Al-Tamimi, Q. A. Al-Haija and K. Alrawashdeh, "Zero-Trust Architecture for Securing Internet of Things (IoT) Networks: A Review," *2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, Veliko Tarnovo, Bulgaria, 2024, pp. 1–6, doi: 10.1109/CIEES62939.2024.10811176.

[8] M. B. Muzammil, M. Bilal, S. Ajmal, S. C. Shongwe and Y. Y. Ghadi, "Unveiling Vulnerabilities of Web Attacks Considering Man in the Middle Attack and Session Hijacking," *IEEE Access*, vol. 12, pp. 6365–6375, 2024, doi: 10.1109/ACCESS.2024.3350444.

[9] F. A. Qazi, "Study of Zero Trust Architecture for Applications and Network Security," *2022 IEEE 19th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, Marietta, GA, USA, 2022, pp. 111–116, doi: 10.1109/HONET56683.2022.10019186.

[10] S. Wang, B. Zhang, B. Shi and Y. Shen, "Analysis and Inspiration of Key Elements of Zero Trust Network Architecture," *2024 2nd International Conference on Mechatronics, IoT and Industrial Informatics (ICMIII)*, Melbourne, Australia, 2024, pp. 938–941, doi: 10.1109/ICMIII62623.2024.00180.

[11] S. Munasinghe, N. Piyarathna, E. Wijerathne, U. Jayasinghe and S. Namal, "Machine Learning Based Zero Trust Architecture for Secure Networking," *2023 IEEE 17th International Conference on Industrial and Information Systems (ICIIS)*, Peradeniya, Sri Lanka, 2023, pp. 1–6, doi: 10.1109/ICIIS58898.2023.10253610.

[12] B. Huber and F. Kandah, "Zero Trust+: A Trusted-based Zero Trust Architecture for IoT at Scale," *2024 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, 2024, pp. 1–6, doi: 10.1109/ICCE59016.2024.10444321.

[13] C. Zhang et al., "Tag-Based Trust Evaluation In Zero Trust Architecture," *2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, Guangzhou, China, 2022, pp. 772–776, doi: 10.1109/IAECST57965.2022.10062213.

[14] National Institute of Standards and Technology (NIST), *Zero Trust Architecture*, NIST Special Publication 800-207, Gaithersburg, MD, USA, 2020.

[15] P. Phiayura and S. Teerakanok, "A Comprehensive Framework for Migrating to Zero Trust Architecture," *IEEE Access*, vol. 11, pp. 19487–19511, 2023, doi: 10.1109/ACCESS.2023.3248622.