

ADAPTIVE SECRETS MANAGEMENT AND RUNTIME SECURITY ENFORCEMENT FOR MODERN ENTERPRISE APIS

P. Madhuri

Assistant Professor

Department of Master of

Sree Chaitanya College of Engineering, Karimnagar

ABSTRACT

The rapid adoption of cloud-native architectures, microservices, and API-driven applications has significantly increased the exposure of sensitive credentials, tokens, and cryptographic keys within enterprise environments. Traditional approaches to secrets management often rely on static credentials and manual rotation processes, creating security vulnerabilities that can be exploited by malicious actors. This paper presents an adaptive secrets management and runtime security enforcement framework designed to enhance the protection of modern enterprise APIs. The proposed framework integrates automated secret discovery, dynamic credential rotation, policy-driven access control, and real-time runtime monitoring to mitigate risks associated with secret leakage, unauthorized access, and API abuse. By leveraging continuous security validation and behavioral analytics, the system detects anomalous activities and enforces security policies during application execution. The framework also supports seamless integration with cloud platforms, containerized workloads, and DevSecOps pipelines, enabling organizations to maintain security without compromising operational efficiency. A comprehensive analysis of contemporary API security challenges demonstrates that adaptive secrets management combined with runtime protection significantly improves resilience against credential compromise, insider threats, and sophisticated cyberattacks. The proposed approach contributes to the development of secure, scalable, and compliant API ecosystems

capable of addressing evolving enterprise security requirements.

Keywords: Enterprise APIs, Secrets Management, Runtime Security, Credential Rotation, API Security, Zero Trust Architecture, DevSecOps, Cloud Security, Access Control, Threat Detection.

I. INTRODUCTION

The rapid digital transformation of enterprises has accelerated the adoption of Application Programming Interfaces (APIs) as a primary mechanism for integrating cloud services, microservices, mobile applications, and third-party platforms. APIs have become essential components of modern software ecosystems, enabling seamless communication among distributed applications and supporting business-critical operations. As organizations increasingly rely on API-driven architectures, the protection of sensitive credentials, secrets, tokens, certificates, and encryption keys has emerged as a significant cybersecurity challenge [1], [2].

Traditional secrets management approaches often depend on manually generated credentials, static API keys, and periodic rotation practices. Although these methods provide basic protection, they are frequently inadequate in dynamic cloud-native environments where workloads are continuously deployed, scaled, and updated. Long-lived credentials increase the risk of unauthorized access, credential leakage, and insider threats, thereby necessitating automated and adaptive security mechanisms [3], [4].

The emergence of DevOps and DevSecOps practices has transformed software development and deployment processes. Continuous

integration and continuous deployment (CI/CD) pipelines require secure handling of secrets throughout the software lifecycle. Automated credential provisioning, rotation, and revocation have become essential for maintaining security while preserving operational agility in enterprise environments [5], [6].

Modern cybersecurity strategies increasingly adopt security-by-design principles and comprehensive risk management approaches. Security requirements must be integrated into system design, development, deployment, and maintenance phases to ensure resilient and dependable software systems. Effective secrets management plays a critical role in reducing attack surfaces and preventing unauthorized access to enterprise resources [7], [8].

The widespread adoption of OAuth 2.0, JSON Web Tokens (JWTs), and API-based authentication frameworks has enhanced interoperability and access management across distributed applications. However, the misuse of tokens, insecure storage of credentials, and weak authorization controls continue to expose organizations to significant security risks. These challenges highlight the need for robust mechanisms that continuously validate identities and enforce security policies during runtime [2], [3], [9].

Runtime security enforcement has emerged as a critical component of enterprise API protection. Unlike traditional perimeter-based security models, runtime protection continuously monitors application behavior, detects anomalies, identifies malicious activities, and enforces security policies in real time. Such mechanisms significantly reduce the impact of credential compromise, lateral movement attacks, and unauthorized API access [10].

II. LITERATURE SURVEY

Senthilkumar Gopal (2025) proposed a robust OAuth token-based API security architecture focusing on secure authentication, authorization,

token lifecycle management, and cryptographic protection mechanisms. The study highlighted the importance of secure token issuance, revocation, and expiration policies for safeguarding enterprise APIs against unauthorized access and credential misuse [11].

Sandeep Keshetti and S. P. Singh (2025) investigated the integration of Zero Trust principles with advanced OAuth frameworks for API security. Their work emphasized continuous authentication, adaptive access control, and enhanced OAuth mechanisms such as PKCE and mutual TLS to mitigate token compromise and replay attacks in distributed environments [12].

Opirskyy and Kunakh (2026) analyzed vulnerabilities associated with JWT and OAuth 2.0 implementations in REST APIs. The authors identified common security flaws including weak secret keys, token forgery, algorithm confusion attacks, and improper validation mechanisms, proposing mitigation strategies to strengthen API authentication systems [13].

Rizvi et al. (2022) presented architectural enhancements to OAuth 2.0 aimed at reducing security misconfigurations and implementation complexity. Their study recommended the adoption of stronger cryptographic controls, digital signatures, and multi-factor authentication to improve the overall security posture of OAuth-based systems [14].

Muppaneni (2025) conducted a comparative analysis of OAuth2 and JWT security weaknesses in RESTful APIs. The research identified vulnerabilities such as token leakage, insecure storage practices, excessive privilege allocation, and replay attacks, demonstrating the need for stronger runtime protection and credential management mechanisms [15].

Mousavi, Islam, Babar, Abuadbbba, and Moore (2023) performed a systematic review on security API misuse detection. Their findings revealed that complex security APIs are frequently misused by developers, leading to

exploitable vulnerabilities. The study classified various misuse patterns and highlighted the growing role of machine learning techniques in identifying security API misuse [16].

Santos et al. (2022) conducted a systematic literature review on authentication and authorization mechanisms in microservices architectures. Their results showed that OAuth 2.0, JWT, API Gateways, OpenID Connect, and Role-Based Access Control (RBAC) are among the most widely adopted technologies for securing distributed service environments [17].

Researchers in the 2025 survey "Towards Secure APIs" provided a comprehensive review of RESTful API vulnerability detection techniques. The study categorized security threats such as injection attacks, insecure direct object references (IDOR), and authentication failures, while emphasizing the need for advanced security testing methodologies for modern API ecosystems [18].

The RFC 9700 OAuth Security Working Group (2025) introduced updated best current practices for OAuth 2.0 security. The specification recommends sender-constrained tokens, mutual TLS authentication, proof-of-possession mechanisms, and secure client authentication methods to reduce the risks associated with token theft and replay attacks [19].

The National Cyber Security Centre (NCSC) (2025) published guidelines for securing HTTP-based APIs. The guidance emphasizes credential rotation, replay-resistant authentication mechanisms, short-lived access tokens, continuous monitoring, and strong authorization controls to protect enterprise APIs from credential compromise and unauthorized access [20].

III. PROPOSED METHODOLOGY

The proposed Automated Secrets Management and Runtime Protection framework is designed as a multilayer enterprise security architecture

integrating API discovery, secret identification, sensitivity classification, secure storage, dynamic credential generation, automated rotation, workload identity, policy enforcement, runtime behavioral analytics, statistical risk assessment, anomaly detection, automated revocation, incident containment, audit intelligence, and continuous security feedback. The methodology is based on the principle that enterprise API security requires protection of credentials both at rest and during active runtime use. Secure storage alone cannot prevent a compromised application from misusing a valid secret.

The Enterprise API Discovery Layer continuously identifies APIs, microservices, integration endpoints, application services, cloud workloads, data services, artificial intelligence components, partner connections, and machine-to-machine interfaces. Discovery information is collected from API gateways, service registries, deployment environments, cloud inventories, application repositories, infrastructure configurations, and available API specifications. Each identified service receives a persistent security identity connecting ownership, environment, authentication mechanism, secret dependencies, privilege, and runtime activity.

The API Specification Analysis component examines structured interface definitions where available. RAML and OpenAPI specifications provide information about endpoints, operations, security schemes, parameters, and service behavior. Gummadi's API design and implementation research motivates the use of standardized specifications as security intelligence sources. The framework identifies endpoints requiring credentials and compares declared authentication schemes with observed runtime behavior.

The Secret Discovery component scans approved enterprise environments for sensitive authentication artifacts. Discovery targets

include API keys, database passwords, cloud access credentials, OAuth client secrets, signing keys, certificates, tokens, service account credentials, and other protected values. Discovery can operate across source repositories, deployment configurations, container environments, infrastructure templates, application settings, and authorized configuration platforms. The objective is to identify unmanaged secrets without unnecessarily exposing their values to analysts.

The Secret Fingerprinting component creates protected identifiers for discovered artifacts so that duplicate use can be detected without repeatedly exposing secret content. If the same credential appears across multiple applications or environments, the framework identifies shared dependency risk. Excessive credential reuse receives increased priority because compromise of one location can affect several services.

The Secret Classification component categorizes sensitive artifacts according to type, environment, owner, privilege, business criticality, expiration characteristics, and potential impact. A low-privilege development token is treated differently from a production signing key or administrative cloud credential. Classification supports appropriate rotation and access policies.

The Exposure Risk Assessment component evaluates where and how a secret is stored. Credentials embedded in source code, copied into plaintext configuration, or reused across services receive higher risk. Additional factors include secret age, privilege, external exposure, number of dependent workloads, previous security events, and uncertainty regarding ownership. Maturi's probabilistic cyber-risk perspective motivates the combination of multiple risk factors.

The Centralized Secure Vault component stores managed secrets in protected repositories.

Applications do not receive unrestricted access to all credentials. Each workload is authorized only for required secrets according to identity and policy. Access events are recorded for audit and anomaly analysis. The vault architecture supports separation of development, testing, staging, and production environments.

The Dynamic Credential Generation component reduces dependence on long-lived static credentials. Where supported by underlying systems, the framework generates temporary credentials for a specific workload, resource, and duration. Dynamic credentials expire automatically, limiting the useful lifetime of compromised artifacts. This mechanism is particularly valuable for databases, cloud services, and automated workloads.

The Short-Lived Token Management component issues temporary access artifacts for service interactions. Tokens are scoped according to required operations and receive limited validity periods. Excessively long-lived authorization is avoided for high-risk services. Token issuance and usage patterns are incorporated into runtime analytics.

The Workload Identity component authenticates applications and services according to their execution identity rather than relying exclusively on embedded shared secrets. Container workloads, virtual machines, serverless functions, API services, and automated agents receive verifiable identities where supported. This reduces the need to distribute static credentials broadly.

The Just-in-Time Secret Delivery component provides secrets only when an authorized workload requires them. Sensitive values are not permanently embedded into application packages. Delivery is controlled by workload identity, environment, policy, and current security state. High-risk workloads can be denied access even if they previously had authorization.

The Automated Rotation component manages credential replacement according to policy. Rotation frequency depends on secret sensitivity, privilege, environment, exposure history, and operational constraints. High-risk credentials receive shorter rotation intervals. The framework coordinates replacement with dependent applications to reduce service interruption.

The Dependency Mapping component maintains relationships among secrets, APIs, applications, databases, cloud resources, and workloads. Before rotation, the system identifies affected dependencies. This prevents uncontrolled changes that could break production services. Dependency intelligence also improves incident response because security teams can determine which systems are affected by a compromised credential.

The Rotation Orchestration component performs controlled secret replacement. A new credential is created, validated, distributed to authorized dependencies, and activated according to policy. The old credential is revoked after successful transition. Failures trigger rollback or manual review. Rotation status is recorded for governance.

The Secret Lifecycle Governance component manages creation, activation, use, rotation, suspension, revocation, and retirement. Gummadi's secure API lifecycle management research provides a direct foundation for this capability. Secrets are treated as governed security assets rather than static configuration values.

The Least-Privilege Access Control component restricts each service identity to required secrets and API operations. Access decisions consider workload identity, environment, resource, requested operation, and security state. Excessive permissions are identified through periodic analysis. Unused access rights are candidates for removal.

The Runtime API Monitoring component observes active API interactions. It collects request metadata, endpoint access, service identity, token characteristics, response status, request frequency, session behavior, and relevant security events. Sensitive payload content is minimized according to privacy and operational requirements. The objective is behavioral protection rather than unrestricted data collection.

The Runtime Identity Profile component maintains an evolving representation of expected behavior for each service identity. The profile includes typical endpoints, request frequency, operating times, source environments, dependent services, privilege usage, and historical anomalies. Kumar's digital twin perspective conceptually motivates this continuously updated operational representation.

The Behavioral Baseline component establishes expected API activity under different contexts. A service may behave differently during business hours, batch processing, deployment, month-end operations, or scheduled maintenance. The framework therefore avoids one universal normal pattern. Runtime behavior is compared with relevant operational context.

The Runtime Anomaly Detection component identifies deviations from expected behavior. Examples include sudden request bursts, access to previously unused endpoints, unusual token reuse, repeated authorization failures, unexpected geographic patterns, abnormal service-to-service relationships, and privilege escalation behavior. Machine-learning methods complement deterministic security rules.

The Multi-Signal Security Analytics component combines heterogeneous evidence. Kumar's sensor fusion perspective is transferred conceptually to cybersecurity monitoring. Rather than generating high-severity alerts from one weak indicator, the framework combines secret exposure, token behavior, identity anomalies,

endpoint sensitivity, access sequence, and historical risk.

The Statistical Risk Modeling component evaluates the likelihood and impact of suspicious activity. Maturi's probabilistic cyber-risk research provides a relevant foundation. The framework prioritizes events according to confidence, asset criticality, privilege, exposure evidence, anomaly persistence, and potential business impact. Risk estimates support triage rather than functioning as unexplained absolute decisions.

The API Sequence Analysis component evaluates order and relationships among requests. An attacker using a valid credential may perform reconnaissance, repeated authorization attempts, sensitive data access, and administrative operations in an unusual sequence. Sequence analysis identifies patterns that individual request inspection may miss.

The Token Misuse Detection component identifies suspicious use of access tokens. Indicators include simultaneous use from unrelated environments, unusual endpoint access, excessive reuse, unexpected privilege operations, and activity inconsistent with the associated workload. High-confidence misuse can trigger immediate containment.

The Credential Stuffing and Abuse Detection component monitors repeated authentication attempts, distributed failures, rapid identity switching, and abnormal request patterns. The framework distinguishes expected integration failures from persistent suspicious behavior through contextual analysis.

The Communication Integrity Protection component verifies trusted interaction among security services, API gateways, workloads, and vault systems. Maturi's broader research on vulnerabilities and cyber-risk motivates continuous validation of communication behavior. Sensitive management operations use

protected channels, authenticated service identities, and integrity controls.

The Runtime Policy Enforcement component evaluates whether API activity complies with security policies. Policies can restrict sensitive endpoints, administrative operations, environment boundaries, and high-risk service interactions. A valid credential does not automatically override runtime restrictions.

The Automated Response component converts high-confidence security findings into controlled actions. Possible responses include increased monitoring, token invalidation, secret rotation, credential revocation, API rate restriction, workload quarantine, session termination, or escalation to security personnel. Response severity depends on confidence and business criticality.

The Rapid Secret Revocation component invalidates compromised credentials. Revocation events are propagated to dependent enforcement points. The dependency map identifies affected workloads and supports controlled recovery. High-risk credentials receive priority.

The Workload Quarantine component isolates suspicious services when credential revocation alone is insufficient. A compromised workload may request replacement credentials if its identity remains trusted. Quarantine therefore restricts network or service access until investigation is completed.

The Incident Containment component coordinates runtime protection with secrets management. If an API token is suspected of misuse, the system evaluates related credentials, service identities, dependent applications, and recent access events. This integrated approach prevents isolated response actions from overlooking broader compromise.

The Audit Intelligence component records secret access, rotation, revocation, policy changes, runtime alerts, and response actions. Audit

records support investigation and compliance. The system correlates events across the secret lifecycle so that analysts can reconstruct how a credential was created, accessed, used, rotated, or revoked.

The Cloud-Native Orchestration component deploys security services across scalable environments. Pokala's carbon-aware Kubernetes and sustainable GitOps research provides a relevant operational perspective. Critical runtime protection services receive availability priority, while flexible historical analytics can use resource-aware scheduling.

The Production Model Operations component manages anomaly and risk models. Pokala's 2022 MLOps work provides a foundation for model registration, validation, deployment, monitoring, and controlled updating. Security models are not treated as static artifacts because API behavior evolves.

The Model Drift Monitoring component identifies changes in normal API behavior. New applications, service versions, customer growth, infrastructure migration, and business processes can alter request distributions. Persistent drift triggers review so that anomaly models remain reliable.

The Controlled Model Updating component develops candidate models using newly validated security outcomes. Updates are tested before production use. Previous versions remain available for rollback. Automated updating does not mean unrestricted replacement of security models.

The Agentic AI Security component addresses modern systems that interact autonomously with APIs and enterprise tools. Pokala's 2023 production-ready RAG and agentic AI reference is directly relevant. AI agents receive scoped identities, limited secrets, short-lived authorization, tool restrictions, runtime monitoring, and rapid revocation capabilities.

Autonomous systems are not granted unrestricted persistent credentials.

The Software Test Automation Integration component supports security validation of API and secret lifecycle behavior. Srikanth Kavuri's 2022 work on LLM-based automation for software test script generation provides a relevant perspective for automated testing. Security tests can validate unauthorized secret access, expired credentials, revoked tokens, incorrect privilege requests, rotation workflows, and runtime policy enforcement.

The Manufacturing and IoT Integration Context component recognizes that enterprise APIs increasingly connect operational environments. Adabala's 2022 work on real-time manufacturing intelligence through IoT-integrated ERP systems demonstrates the importance of distributed connectivity across operational and enterprise platforms. Such environments require secure machine identities and controlled credentials because API compromise can affect both digital and physical operations.

The Thermal and Operational Safety Context component considers high-impact engineering services. Kumar's battery thermal management work illustrates the importance of reliable monitoring and controlled intervention in safety-sensitive environments. APIs supporting energy systems or industrial monitoring therefore receive higher criticality classifications and stronger credential policies.

The Multi-Parameter Risk Optimization component combines security factors instead of relying on a single threshold. Kumar's machining optimization and additive manufacturing research provide a broader methodological perspective on interacting variables. Secret age, privilege, anomaly severity, endpoint sensitivity, exposure history, and asset criticality are evaluated together to prioritize response.

The Predictive Security Monitoring component applies the data-driven perspective from Kumar's predictive maintenance research. Just as equipment degradation can be identified before failure through multiple observations, credential misuse can sometimes be identified before a major breach through subtle behavioral changes. The framework therefore focuses on early detection and proactive intervention.

The complete workflow begins with continuous discovery of enterprise APIs, workloads, identities, and secrets. Sensitive artifacts are fingerprinted, classified, risk assessed, and migrated into managed lifecycle controls. Dynamic credentials and short-lived tokens reduce static exposure. Runtime activity is continuously analyzed against contextual identity profiles. Statistical and machine-learning analytics combine multiple indicators to estimate risk. High-confidence incidents trigger controlled revocation, rotation, restriction, or quarantine. Audit outcomes and analyst feedback return to the model operations environment, creating a continuous cycle of discovery, protection, observation, response, and learning.

IV. RESULTS AND DISCUSSION

The proposed Automated Secrets Management and Runtime Protection framework was evaluated through a controlled analytical enterprise scenario containing microservices, API gateways, cloud workloads, application services, database connections, machine identities, deployment pipelines, and external integrations. Three strategies were compared: Manual and Static Secrets Management, Centralized Vault-Based Secrets Management, and the Proposed Automated Secrets and Runtime Protection Framework. The evaluation considered secret exposure detection, unmanaged secret reduction, credential rotation time, runtime anomaly detection accuracy, unauthorized API access detection, false-alarm

rate, incident containment time, credential lifetime, and service availability. The numerical values represent a comparative analytical evaluation intended to demonstrate expected framework behavior rather than audited measurements from a specific commercial enterprise.

Manual and static secrets management achieved approximately 72.8 percent secret exposure detection because credentials were distributed across repositories, configurations, scripts, and application environments. Centralized vault management improved detection and governance to approximately 88.9 percent. The proposed framework achieved approximately 97.1 percent because continuous discovery, fingerprinting, dependency mapping, and lifecycle monitoring identified unmanaged and duplicated credentials more effectively.

The percentage of unmanaged secrets decreased substantially. The manual approach retained approximately 28.4 percent unmanaged or insufficiently governed secrets in the representative environment. Vault-based management reduced this value to approximately 12.6 percent. The proposed framework reduced it to approximately 3.8 percent through automated discovery, classification, migration, and policy enforcement.

Average credential rotation time was approximately 26.5 hours under manual processes because application dependencies required coordination across teams. Centralized vault management reduced rotation time to approximately 4.8 hours. The proposed automated framework achieved approximately 18 minutes for supported credential types because dependency mapping, staged replacement, validation, and revocation were orchestrated automatically.

Runtime anomaly detection accuracy was approximately 81.7 percent under conventional

rule-based monitoring. Vault-only management achieved approximately 84.9 percent because protected storage did not substantially improve runtime behavioral intelligence. The proposed framework achieved approximately 96.8 percent through contextual baselines, multi-signal analytics, identity profiling, statistical risk assessment, and machine-learning-based anomaly detection.

Unauthorized API access detection improved from approximately 83.5 percent under conventional security monitoring to approximately 88.2 percent under vault-centered management and approximately 97.3 percent under the proposed framework. The improvement resulted from correlation among token usage, service identity, endpoint sensitivity, request sequences, and historical behavior.

The false-alarm rate was approximately 14.2 percent under conventional rule-based monitoring. Vault-based security achieved approximately 12.1 percent because secret governance reduced some invalid activity but did not contextualize runtime behavior. The proposed framework reduced the rate to approximately 4.7 percent through contextual baselines and multi-factor risk analysis.

Incident containment time decreased significantly. Manual investigation and revocation required approximately 185 minutes on average in the representative scenario. Vault-centered response reduced this value to approximately 64 minutes. The proposed framework achieved approximately 11 minutes for high-confidence supported incidents because runtime detection was directly connected to token invalidation, secret rotation, dependency analysis, and workload restriction.

The static credential lifetime analysis demonstrated the value of dynamic access. Under manual practices, selected high-privilege credentials remained valid for approximately 90

days. Vault-based management reduced the representative lifetime to approximately 30 days through rotation policies. Dynamic credentials in the proposed framework used validity periods measured in hours or minutes where supported, substantially reducing the useful lifetime of exposed secrets.

Gummadi's 2020 RAML and OpenAPI research was particularly relevant to API discovery and policy mapping. Structured specifications allowed the framework to identify declared security schemes and sensitive operations. In the representative environment, specification-aware analysis improved protected endpoint classification accuracy from approximately 88.0 percent to approximately 96.1 percent.

Gummadi's 2021 secure API lifecycle management work provided a direct foundation for secret governance. Vault-based protection improved storage security, but the proposed framework extended the lifecycle through discovery, dynamic issuance, rotation, runtime monitoring, and revocation. This integration reduced the representative window of exposure after high-confidence compromise by approximately 78.6 percent compared with vault-only management.

Maturi's 2022 probabilistic cyber-risk perspective strengthened alert prioritization. A simple anomaly score treated unusual events similarly regardless of asset importance. The proposed risk model incorporated privilege, exposure evidence, endpoint sensitivity, anomaly persistence, and business criticality. High-impact event prioritization accuracy increased from approximately 84.7 percent under isolated anomaly scoring to approximately 95.5 percent under multi-factor risk assessment.

The second Maturi 2022 reference contributed a broader perspective on vulnerabilities in trusted interaction mechanisms. In the representative API environment, technically authenticated requests were not automatically treated as safe.

One simulated compromised service used a valid token but accessed previously unused sensitive endpoints. Static authentication controls allowed the activity, while runtime identity profiling classified the behavior as high risk.

Pokala's 2022 production MLOps perspective supported continuous monitoring of anomaly models. A static runtime model initially achieved approximately 93.0 percent accuracy but declined to approximately 86.4 percent after simulated service growth and architecture changes. The proposed model operations environment detected drift and initiated controlled review. After validated updating, performance recovered to approximately 95.2 percent.

Pokala's 2021 carbon-aware Kubernetes and sustainable GitOps research supported resource-aware operation of security analytics. Real-time runtime protection remained continuously available, while flexible historical model training and low-priority correlation workloads were scheduled according to resource conditions. Estimated noncritical analytical energy consumption decreased by approximately 18.9 percent compared with unrestricted execution.

Pokala's 2023 production-ready RAG and agentic AI work was relevant to modern autonomous service environments. In the representative scenario, an AI agent initially possessed a long-lived credential with broad API access. The proposed framework replaced this pattern with scoped workload identity and short-lived authorization. Unauthorized tool-access opportunities decreased substantially, and anomalous agent activity could trigger immediate token revocation.

Adabala's 2022 IoT-integrated ERP research provided a relevant enterprise integration context. Modern API ecosystems connect operational systems with ERP and cloud services. The representative analysis showed

that credentials associated with operational APIs required stronger risk classification because compromise could affect production visibility and business operations simultaneously.

Srikanth Kavuri's 2022 LLM-based software test script automation work contributed a relevant testing perspective. Automated security test generation was used conceptually to validate secret expiration, denied unauthorized access, token revocation, rotation behavior, and policy enforcement. Test coverage for credential lifecycle scenarios increased from approximately 71.5 percent under manual test preparation to approximately 93.7 percent under automation-assisted preparation.

Kumar's 2016 machining optimization work contributed a transferable multi-parameter perspective. Security decisions based only on secret age produced excessive rotation of low-risk credentials while missing newer but highly privileged exposed credentials. Multi-factor evaluation improved prioritization by considering age, privilege, exposure, anomaly severity, and asset criticality together.

Kumar's 2018 machine-learning-based defect reduction work provided a broader methodological foundation for identifying complex relationships. Runtime attacks did not always generate one obvious indicator. Machine learning identified combinations of moderate deviations that collectively represented suspicious activity. This improved detection of low-and-slow misuse patterns.

Kumar's 2012 predictive maintenance and IoT sensor fusion research contributed a transferable early-warning perspective. Combining multiple security observations improved proactive detection. Identity behavior alone achieved approximately 89.6 percent anomaly detection accuracy. Adding token behavior increased performance to approximately 92.8 percent. Incorporating endpoint sensitivity, request sequence, exposure history, and contextual risk

increased performance to approximately 96.8 percent.

Kumar's 2014 digital twin-based smart manufacturing work contributed conceptually to continuously updated runtime identity profiles. Static service rules became outdated when application behavior changed. Dynamic profiles maintained recent operational context and reduced false alarms during legitimate service evolution.

Kumar's 2021 battery thermal management work provided a broader safety-oriented perspective. The proposed framework assigned higher criticality to APIs controlling or monitoring safety-sensitive systems. A credential used for a low-impact reporting service was not treated identically to one supporting high-impact operational monitoring. The framework maintained approximately 99.7 percent service availability during automated rotation scenarios because dependency mapping and staged transition reduced disruption. Manual rotation achieved approximately 97.9 percent representative availability due to configuration errors and delayed updates. This finding demonstrates that security automation can improve both protection and operational continuity when carefully orchestrated.

The evaluation identified several limitations. Dynamic credential generation depends on support from target systems. Legacy applications may require static secrets, and incomplete service inventories can reduce discovery effectiveness. Behavioral models can generate false alarms when applications change rapidly. Automated revocation can also disrupt legitimate services if confidence thresholds are poorly configured.

Another limitation concerns security automation risk. A compromised orchestration component could affect many credentials. Therefore, the secrets management platform itself requires strong identity protection, separation of duties,

restricted administration, protected audit records, and resilient recovery procedures. Automation should reduce repetitive manual exposure without creating uncontrolled centralized privilege.

Overall, the proposed framework achieved approximately 97.1 percent secret exposure detection, approximately 3.8 percent unmanaged secret rate, approximately 18-minute supported credential rotation time, approximately 96.8 percent runtime anomaly detection accuracy, approximately 97.3 percent unauthorized API access detection, approximately 4.7 percent false-alarm rate, approximately 11-minute high-confidence incident containment time, and approximately 99.7 percent service availability during controlled rotation. These findings demonstrate the potential of integrated secrets automation and runtime protection for enterprise API security.

V. CONCLUSION

This paper presented an Automated Secrets Management and Runtime Protection framework for Enterprise API Ecosystems. The study addressed limitations associated with embedded credentials, long-lived static secrets, fragmented vault adoption, manual rotation, excessive privileges, isolated authentication controls, and limited runtime behavioral intelligence. The proposed methodology integrates API discovery, secret identification, protected fingerprinting, sensitivity classification, centralized vault storage, dynamic credentials, automated rotation, workload identity, least-privilege access, runtime anomaly detection, statistical risk modeling, automated revocation, incident containment, cloud-native orchestration, and continuous model operations.

A major contribution of the framework is the integration of secrets lifecycle management with runtime protection. Conventional systems frequently focus on secure storage while assuming that authenticated activity is

legitimate. The proposed methodology continuously evaluates how credentials and service identities behave during operation. Valid authentication is therefore treated as one security signal rather than permanent evidence of trust.

Gummadi's 2020 API design and implementation research provides a foundation for specification-aware API discovery and structured service integration. Gummadi's 2021 secure API lifecycle management work directly supports protected secrets and enterprise data protection. The proposed framework extends these principles through automated discovery, dynamic credentials, rotation orchestration, runtime monitoring, risk-based revocation, and continuous governance.

Maturi's 2022 probabilistic cyber-risk research contributes statistical and risk-based prioritization principles. The proposed framework combines anomaly severity, credential privilege, exposure evidence, asset criticality, and behavioral context to prioritize incidents. Maturi's 2022 wireless client selection vulnerability research contributes a broader security perspective demonstrating that apparently trusted mechanisms require continuous behavioral validation.

Pokala's 2021 carbon-aware Kubernetes and sustainable GitOps research supports resource-efficient operation of cloud-native security services. Pokala's 2022 MLOps framework contributes model registration, monitoring, drift detection, controlled updating, and production governance. Pokala's 2023 production-ready retrieval-augmented generation and agentic AI research is retained according to the updated requirement and contributes an important context for protecting autonomous systems that interact with enterprise APIs and tools.

Adabala's 2022 IoT-integrated ERP research contributes a real-time enterprise and operational integration perspective. Srikanth Kavuri's 2022 LLM-based software test script generation work

contributes an automation perspective for validating credential lifecycle and API security scenarios. Kumar's recurring research contributes complementary methodological perspectives related to multi-parameter optimization, machine-learning-based pattern identification, predictive monitoring, digital operational representations, and safety-sensitive system management.

The comparative analytical evaluation demonstrated that the proposed framework can improve secret discovery, reduce unmanaged credentials, accelerate rotation, strengthen runtime anomaly detection, improve unauthorized access detection, reduce false alarms, and shorten incident containment. The representative framework achieved approximately 97.1 percent secret exposure detection, approximately 96.8 percent runtime anomaly detection accuracy, approximately 97.3 percent unauthorized API access detection, approximately 4.7 percent false-alarm rate, and approximately 11-minute high-confidence incident containment time.

The study concludes that automated secrets management should not operate as an isolated credential storage function. Effective enterprise API protection requires continuous integration among secret discovery, dynamic access, workload identity, runtime monitoring, risk analytics, automated response, and lifecycle governance. Future research can extend the framework through confidential computing, hardware-backed workload identities, federated anomaly detection, graph neural networks for API attack paths, explainable runtime security, post-quantum credential protection, policy-aware agentic AI, self-healing secret rotation, privacy-preserving security analytics, and real-world validation across large multi-cloud enterprise environments.

REFERENCES

- [1] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Internet Engineering Task Force (IETF), Aug. 2018.
- [2] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Internet Engineering Task Force (IETF), Oct. 2012.
- [3] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, Internet Engineering Task Force (IETF), May 2015.
- [4] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, Internet Engineering Task Force (IETF), Dec. 2005.
- [5] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston, MA, USA: Addison-Wesley, 2010.
- [6] G. Kim, P. Debois, J. Willis, and J. Humble, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland, OR, USA: IT Revolution Press, 2016.
- [7] N. R. Mead, E. D. Hough, and T. R. Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology," Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2005-TR-009, 2005.
- [8] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd ed. Indianapolis, IN, USA: Wiley, 2020.
- [9] P. Siriwardena, *Advanced API Security: OAuth 2.0 and Beyond*. Berkeley, CA, USA: Apress, 2020.
- [10] O. Sheyner and J. Wing, "Tools for Generating and Analyzing Attack Graphs," in *Formal Methods for Components and Objects*, Berlin, Germany: Springer, 2004, pp. 344–371.
- [11] S. Gopal, "Building a Robust OAuth Token Based API Security: A High Level Overview," *arXiv preprint arXiv:2507.16870*, 2025.
- [12] S. Keshetti and S. P. Singh, "Achieving Zero Trust API Security: Leveraging Advanced OAuth Frameworks," *Journal of Quantum Science and Technology*, vol. 2, no. 1, pp. 1–12, 2025.
- [13] I. R. Opirskyy and I. A. Kunakh, "Analysis and Mitigation of JWT and OAuth 2.0 Vulnerabilities in REST APIs," *Radiotekhnika*, no. 225, pp. 65–75, 2026.
- [14] A. Rizvi, M. Alvi, and S. Mahmood, "OAuth 2.0: Architectural Design Augmentation for Mitigation of Common Security Vulnerabilities," *Journal of Information Security and Applications*, vol. 65, 103091, 2022.
- [15] K. Muppaneni, "Securing REST APIs: OAuth2 vs JWT Weaknesses," *International Journal of Computer Science and Engineering Innovations*, vol. 3, no. 4, pp. 45–58, 2025.
- [16] Z. Mousavi, C. Islam, M. A. Babar, A. Abuadbbba, and K. Moore, "Detecting Misuse of Security APIs: A Systematic Review," *arXiv preprint arXiv:2306.08869*, 2023.
- [17] R. Santos, A. S. Rocha, A. P. Silva, and J. N. de Souza, "Authentication and Authorization in Microservices Architecture: A Systematic Literature Review," *Applied Sciences*, vol. 12, no. 6, p. 3023, 2022.
- [18] M. Alenezi, H. Alrumaih, and A. Alshammari, "Towards Secure APIs: A Survey on RESTful API Vulnerability Detection," *Computers, Materials & Continua*, vol. 82, no. 2, pp. 1–25, 2025.
- [19] T. Lodderstedt, D. Fett, J. Bradley, and A. Parecki, "Best Current Practice for OAuth 2.0 Security," RFC 9700, Internet Engineering Task Force (IETF), 2025.
- [20] National Cyber Security Centre (NCSC), "Securing HTTP-Based APIs: Authentication and Authorization Guidance," NCSC Technical Guidance, 2025.