JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2025, VOL 8, NO S12

Automating YAML Encryption And Secure Access With Ephemeral Credentials

Raman Vasikarla

SRE at Cisco, USA

Abstract

Multi-cloud infrastructure deployments are increasingly reliant on YAML-based configuration management for declarative system provisioning. Sensitive and private credentials that are embedded in configuration files make security a major concern when these files when stored as plaintext or managed through manual processes. Traditional secrets management techniques fail to address the dual challenges of configuration encryption and runtime access control simultaneously. The article presents an integrated framework combining automated YAML encryption with dynamic ephemeral credential generation to eliminate static credential exposure. Hierarchical encryption pipelines employ asymmetric cryptography to separate encryption responsibilities from decryption privileges while maintaining configuration readability. Validation mechanisms enforce cryptographic consistency across deployment environments through automated schema verification and encryption coverage analysis. Ephemeral credential systems implement vault-based dynamic secret generation where access tokens exist only during specific operational tasks. Role generators issue time-limited credentials bound to requesting identities with minimum required privileges. Integration with continuous deployment pipelines occurs through plugin architectures that intercept configuration files at multiple workflow stages. Cross-platform orchestration patterns facilitate the normalization of secret access that is compatible with different cloud environments through unified abstraction layers. The framework is a means of implementing Zero Trust concepts by, among other things, ensuring that there is continuous verification as well as microsegmentation through the use of granular access controls. The full observability integration also supports the recording of encryption operations, credential issuance events, and access patterns for compliance automation. The combined architecture demonstrates how cryptographic automation and dynamic credential management address fundamental weaknesses in contemporary Infrastructure as Code security practices.

Keywords: YAML Encryption, Ephemeral Credentials, Zero Trust Architecture, Dynamic Secret Generation, Infrastructure as Code Security, Multi-Cloud Credential Management

Introduction

The proliferation of cloud-native architectures has substantially changed infrastructure management practices. Infrastructure as Code (IaC) has become the major model for declarative system configuration. YAML files are the primary means of specifying infrastructures, application parameters, and service dependencies across deployment pipelines. The centralization of configuration data creates critical security challenges when sensitive credentials must be embedded within these human-readable structures. Data breach

incidents have escalated significantly across global digital infrastructures. Research analyzing breach patterns reveals that credential exposure through configuration files represents a persistent vulnerability vector [1]. The universal use of YAML as a configuration standard has led to an indirect increase in the potential for credential exposure through various version control systems, artifact repositories, and deployment pipelines. Secrets management in YAML configurations that are based on traditional methods involves placing the secrets in plaintext with access restrictions, manually encrypting with shared keys, or post-deployment secret injection through environment variables. Each method introduces distinct vulnerabilities. Plaintext storage exposes credentials in version control systems. Analysis of data breach incidents demonstrates that inadequate access controls and unencrypted sensitive data constitute primary factors in security compromises [1]. Manual encryption creates key distribution problems requiring synchronization across distributed systems. Environment variable injection complicates configuration reproducibility across environments while introducing timing vulnerabilities during container initialization phases. The persistence of static, long-lived credentials further amplifies risk. Compromised credentials remain valid indefinitely unless manually revoked. Statistical examination of security incidents reveals that cloud infrastructure breaches frequently involve exploitation of static credentials with excessive permissions and extended validity periods [1].

Current research in cloud security emphasizes ephemeral access patterns as fundamental to Zero Trust architectures. Zero Trust represents a paradigm shift from traditional perimeter-based security models to identity-centric access control. The architecture mandates continuous verification of operational context and explicit authorization for every access request [2]. Access credentials should be dynamically generated with the minimum privileges necessary for specific operations. Automatic expiration occurs after task completion within strictly defined time windows [2]. However, existing implementations often treat configuration encryption and runtime credential management as separate concerns. This fragmentation results in security models that introduce operational complexity. Organizations implementing isolated encryption and credential management solutions face integration overhead and inconsistent policy enforcement across deployment environments. The integration gap between encrypted configuration storage and dynamic credential issuance represents a critical weakness in contemporary IaC security practices [1].

Zero belief structure eliminates implicit agreement based on network vicinity. Every access attempt requires authentication and authorization evaluation regardless of origin [2]. Traditional security models assume trust within network perimeters. Zero Trust fundamentally challenges this assumption by requiring verification at every transaction. The model emphasizes least-privilege access principles where permissions remain scoped to immediate operational requirements. Dynamic policy enforcement adapts access controls based on contextual factors, including user identity, device posture, and resource sensitivity [2].

This research addresses these limitations by presenting a unified framework. The framework automates YAML encryption through validation-enforced pipelines while simultaneously implementing ephemeral credential generation for runtime access. The contribution extends beyond individual security mechanisms. Encryption methods that are automatically triggered, dynamically generated secrets, and observability-driven validation are all integrated at the architectural level. A deep integration like this is a step towards the goal of secure, scalable, and compliant multi-cloud deployments. Automated encryption coverage validation analyzes configuration parameters against sensitive data patterns. Manual review requirements decrease while consistent cryptographic protection spans development, staging, and production environments [1].

Related Work and Methodology

Existing literature addresses configuration encryption and credential management as isolated security domains. Prior frameworks focus predominantly on single-aspect solutions: encryption tools for static configuration protection or secret management platforms for credential storage. Research on Infrastructure as Code security emphasizes either pre-

deployment validation mechanisms or runtime access control systems without architectural integration between these complementary domains. Zero Trust implementations in cloud environments typically concentrate on network segmentation and identity verification while overlooking configuration-level security controls.

The article addresses the integration gap by presenting a unified framework combining hierarchical YAML encryption with dynamic ephemeral credential generation. The methodology implements asymmetric cryptography for configuration protection through selective parameter encryption, preserving operational transparency while securing sensitive data. Validation pipelines enforce cryptographic consistency using automated schema verification and coverage analysis across deployment environments. Vault-based secret engines generate time-limited credentials bound to specific operational contexts, eliminating static credential persistence. Role generators define privilege scopes and temporal constraints, implementing least-privilege access patterns at granular service levels.

The framework integrates with continuous deployment pipelines through plugin architectures operating at pre-commit, build-time, and deployment stages. Cross-platform abstraction layers normalize secret access across heterogeneous cloud environments. Observability integration captures encryption operations, credential lifecycle events, and access patterns for compliance automation. The architectural contribution demonstrates operational integration of configuration encryption with runtime credential management, advancing Infrastructure as Code security beyond isolated control implementations.

YAML Encryption Framework Architecture

Hierarchical Encryption Pipeline Design

The encryption framework implements a hierarchical approach to YAML security. Asymmetric cryptography separates encryption responsibilities from decryption privileges. The Hiera-Eyaml encryption tool provides role-based encryption where specific key-value pairs within YAML structures receive targeted encryption. This approach preserves file readability while securing sensitive parameters. Selective encryption maintains compatibility with version control systems and human review processes. Non-sensitive configuration remains in plaintext to facilitate collaborative development workflows.

Encryption keys follow a tiered management structure. Public keys enable widespread encryption capabilities across development teams. Private decryption keys remain restricted to deployment automation systems. Cryptographic security relies on mathematical hardness assumptions underlying public key systems. The computational intractability of certain problems forms the foundation for asymmetric encryption schemes [3]. Key rotation occurs through automated pipelines that generate new key pairs. The pipelines re-encrypt affected configurations and deprecate expired keys according to predefined lifecycle policies. Automated key lifecycle management eliminates manual intervention requirements.

The separation of encryption and decryption authorities prevents credential exposure during configuration authoring. Deployment systems maintain necessary access without exposing decryption capabilities to broader organizational roles. Provable security in cryptographic systems depends on rigorous mathematical foundations. Security proofs demonstrate that breaking encryption schemes requires solving computationally infeasible problems [3]. Asymmetric encryption provides guarantees for confidentiality through mathematical properties. Public key distribution enables anyone to encrypt data. Only entities possessing corresponding private keys can decrypt protected information. This cryptographic property proves valuable in collaborative development environments where multiple teams contribute configuration changes.

Validation and Consistency Enforcement

Pre-deployment validation pipelines enforce cryptographic consistency across configuration files. Automated schema verification and encryption coverage analysis detect configuration defects before deployment. Validation rules identify missing encryption on parameters matching sensitive data patterns. Connection strings, authentication tokens, and cryptographic material require mandatory encryption enforcement. Regular expression patterns combined

with semantic analysis identify potential secrets. Parameter naming conventions and value structures provide indicators for sensitive data classification.

Cloud security frameworks face significant implementation challenges. Analysis of contemporary frameworks reveals common problems, including incomplete security controls and inadequate validation mechanisms [4]. Security misconfigurations represent leading causes of cloud infrastructure vulnerabilities. Automated validation reduces human error by enforcing encryption policies consistently across configuration artifacts. The validation framework implements multi-layered detection mechanisms. Syntactic analysis examines parameter names for keywords indicating sensitive content. Semantic analysis evaluates value patterns to identify credential formats.

Cross-environment consistency checks verify that encrypted parameters exist across development, staging, and production configurations. Configuration drift between environments creates deployment failures and security vulnerabilities. Cloud security frameworks must address multi-tenancy concerns and access control complexities [4]. Hash-based validation ensures encrypted values match expected formats without exposing plaintext content. Cryptographic hash functions generate unique fingerprints for encrypted parameters. The validation system compares hash values across environments to confirm configuration consistency. Automated testing of encryption integrity occurs throughout the deployment pipeline. Pre-commit hooks prevent unencrypted secrets from entering version control. Build-time validation confirms encryption coverage before artifact creation. Deployment-time checks verify decryption capabilities before releasing configurations to production systems [4].

Table 1. YAML Encryption Framework Components [3, 4]

Framework Component	Security Function
Hierarchical Encryption Pipeline	Implements asymmetric cryptography, separating encryption responsibilities from decryption privileges across development and deployment teams
Selective Parameter Encryption	Targets specific key-value pairs within YAML structures while preserving file readability for non-sensitive configuration
Tiered Key Management	Public keys enable widespread encryption capabilities, while private decryption keys remain restricted to deployment automation systems
Automated Key Rotation	Generates new key pairs, re-encrypts affected configurations, and deprecates expired keys according to predefined lifecycle policies
Pre-deployment Validation	Enforces cryptographic consistency through automated schema verification and encryption coverage analysis
Pattern-based Detection	Identifies potential secrets using regular expressions and semantic analysis of parameter naming conventions and value structures
Cross-environment Consistency	Verifies encrypted parameters exist across development, staging, and production configurations to prevent deployment failures
Hash-based Validation	Ensures encrypted values match expected formats without exposing plaintext content throughout the deployment pipeline

Ephemeral Credentials Architecture

Vault-Based Dynamic Secret Generation

The ephemeral credential system implements a dynamic secret generation model. Credentials exist only for the duration of specific operational tasks. Role generators maintain templates defining privilege scopes, resource access patterns, and temporal constraints for different operational personas. When authenticated principals request access, the secret engine generates unique credentials bound to the requesting identity. Time limitations restrict credentials to the minimum duration required for the intended operation.

Multi-cloud environments present significant challenges for credential management. Organizations deploying applications across heterogeneous cloud platforms require unified approaches to secret storage and access control [5]. Dynamic database credentials exemplify credential generation patterns. Connection requests trigger the generation of temporary database users with restricted permissions. The secret engine creates the user account and assigns role-specific privileges. Connection parameters return to the requesting service immediately after credential generation. Upon credential expiration, automated cleanup processes revoke database permissions. Temporary user accounts undergo removal to ensure no persistent access artifacts remain in target systems.

Secret management systems must provide consistent interfaces across diverse infrastructure platforms. Centralized secret stores enable policy-driven access control regardless of the underlying cloud provider [5]. The credential generation process follows strict temporal boundaries. Maximum validity periods prevent credential misuse beyond authorized operational windows. Short-lived credentials limit the exposure window for compromised authentication material. Credential rotation occurs automatically without requiring manual intervention. Unified secret management architectures reduce operational complexity by standardizing credential lifecycle operations across cloud environments [5].

Credential Lifecycle Management

Credential expiration policies enforce automatic revocation based on maximum time-to-live constraints. Renewal mechanisms exist for long-running processes that demonstrate continued authorization. Lease management systems track all issued credentials, their expiration timestamps, and associated audit metadata. This centralized tracking enables rapid revocation across all active credentials when security events require immediate access termination. Identity and access management in cloud environments faces distinct challenges compared to traditional on-premises systems [6].

The credential lifecycle integrates with identity federation systems. Authentication state inherits from organizational identity providers while maintaining separation between authentication and authorization decisions. Cloud identity management must address challenges, including multi-tenancy, dynamic resource provisioning, and distributed authentication [6]. Token-based authentication flows provide the authentication context. Role generators implement fine-grained authorization policies based on operational requirements. This separation enables consistent credential issuance across heterogeneous authentication sources while maintaining centralized access control.

Federation architectures support multiple identity providers through standardized protocols. Single sign-on capabilities reduce authentication friction while maintaining security boundaries. Access control mechanisms in cloud environments must accommodate rapid scaling and resource elasticity [6]. Credential leases contain metadata describing permission scopes, resource access patterns, and temporal constraints. Audit systems capture all credential issuance and renewal events for compliance reporting. Identity management systems must balance security requirements with operational flexibility while supporting diverse authentication mechanisms [6].

Table 2. Ephemeral Credential Lifecycle Operations [5, 6].

Lifecycle Stage	Operational Mechanism
Credential Request	Authenticated principals request access, triggering secret engine evaluation of role-specific templates
Dynamic Generation	Secret engine creates unique credentials bound to the requesting identity, with time limitations for intended operations
Privilege Assignment	Role generators define permission scopes, resource access patterns, and temporal constraints for operational personas
Temporary Account Creation	Database credentials trigger the generation of temporary users with restricted permissions specific to connection

	requirements
Runtime Distribution	Connection parameters return to requesting services immediately following credential generation
Automatic Expiration	Maximum time-to-live constraints enforce automatic revocation upon task completion or timeout occurrence
Cleanup Automation	Revocation processes remove database permissions and temporary user accounts, ensuring no persistent access artifacts
Lease Tracking	Centralized systems monitor all issued credentials, expiration timestamps, and audit metadata for rapid revocation capabilities

Integration and Automation Workflows

Continuous Integration Pipeline Integration

The framework integrates with continuous integration and continuous deployment pipelines through plugin architectures. These architectures intercept configuration files during build processes. Encryption validation occurs as a pre-commit hook. This mechanism prevents unencrypted secrets from entering version control systems. Build-time validation repeats encryption coverage checks. Configuration consistency receives verification before artifact creation. Deployment-time decryption occurs within isolated execution environments where deployment automation possesses the necessary decryption keys.

Open-source CI/CD pipelines face numerous security threats across their operational lifecycle. Attack surfaces span multiple stages, including source code repositories, build environments, and deployment mechanisms [7]. Dynamic secret retrieval mechanisms replace static credential embedding. Services query secret engines during initialization rather than reading credentials from configuration files. Configuration files reference secret paths rather than credential values. Runtime resolution occurs through authenticated API calls to secret management systems. This approach ensures credentials never persist in configuration files or deployment artifacts.

CI/CD security requires addressing vulnerabilities at each pipeline stage. Threat actors exploit weaknesses in dependency management, artifact storage, and deployment configurations [7]. Pipeline security extends beyond simple validation checks. Comprehensive integration coordinates multiple security controls across build, test, and deployment stages. Secret injection occurs at the latest possible moment in deployment workflows. Just-in-time credential retrieval minimizes exposure windows. Deployment systems authenticate to secret management platforms using short-lived tokens. Supply chain attacks targeting CI/CD infrastructure represent escalating security concerns [7].

Cross-Platform Orchestration Patterns

Multi-cloud deployments require platform-specific integration patterns for secret retrieval and credential application. The framework implements abstraction layers that normalize secret access across cloud-native identity systems. Consistent configuration management becomes possible regardless of the underlying platform. Deployment orchestration tools retrieve platform-specific credentials through unified interfaces. Appropriate authentication mechanisms apply based on target environment characteristics.

Container-based cloud environments demand specialized security frameworks. Traditional security approaches prove insufficient for dynamic containerized workloads [8]. Service mesh integration extends ephemeral credentials to inter-service communication. Mutual TLS certificates receive automatic provisioning and rotation through certificate authority integration. Certificate lifecycle automation eliminates manual certificate management. Strict identity verification applies to all service-to-service authentication operations.

Container orchestration platforms require specialized credential injection mechanisms. Secrets must reach application containers without exposure through container images or orchestration manifests. Defensive security frameworks for container environments must

adapt to rapidly changing threat landscapes [8]. Cross-platform credential management demands careful consideration of trust boundaries. Secrets crossing platform boundaries require additional encryption and access controls. Container security frameworks employ active defense mechanisms that respond dynamically to detected threats [8]. Federation protocols enable credential portability while maintaining security isolation between cloud environments.

Table 3. CI/CD Pipeline Integration Controls [7, 8]

Integration Point	Security Control Mechanism
Pre-commit Hooks	Encryption validation prevents unencrypted secrets from entering version control systems
Build-time Validation	Repeats encryption coverage checks, ensuring configuration consistency before artifact creation
Deployment Decryption	Occurs within isolated execution environments possessing the necessary decryption keys
Dynamic Secret Retrieval	Configuration files reference secret paths rather than credential values, enabling runtime resolution
Authenticated API Calls	Services query secret management systems during initialization, replacing static credential embedding
Just-in-time Injection	Secret delivery occurs at the latest possible moment in deployment workflows, minimizing exposure windows
Platform Abstraction	Normalized secret access across cloud-native identity systems, enabling consistent configuration management
Service Mesh Integration	Automatic provisioning and rotation of mutual TLS certificates through certificate authority integration

Security and Compliance Considerations

Zero Trust Architecture Alignment

The combined encryption and ephemeral credential framework directly implements core Zero Trust principles. Continuous verification and least-privilege access form the foundation of the security model. Every credential request undergoes authentication and authorization evaluation. No implicit trust exists based on network location or previous access grants. Time-limited credentials enforce continuous re-verification. Services must periodically demonstrate ongoing authorization requirements to maintain access [9].

Zero Trust architectures fundamentally transform traditional security perimeters. The model assumes breach scenarios where attackers may already possess network access. Verification occurs at every transaction regardless of source location [9]. Microsegmentation capabilities emerge from role-specific credential generation. Each service instance receives credentials scoped exclusively to its operational requirements. This granular access control prevents lateral movement following security breaches. Compromised credentials provide access only to explicitly authorized resources for limited time windows.

Network segmentation alone proves insufficient for modern threat landscapes. Identity-centric access controls replace location-based trust assumptions. Zero Trust principles mandate that all resources remain private by default [9]. Dynamic policy enforcement adapts access decisions based on contextual factors. User identity, device posture, resource sensitivity, and behavioral patterns inform authorization decisions. Policy engines evaluate multiple signals before granting access to protected resources [9].

Observability and Audit Integration

Comprehensive logging captures all encryption operations, credential issuance events, and secret access patterns. Audit trails support compliance verification requirements across regulatory frameworks. Log aggregation systems centralize this telemetry. Security operations teams detect anomalous access patterns and potential credential misuse through centralized

monitoring. Automated alerting triggers incident response workflows when credential issuance patterns deviate from established baselines [10].

Observability in distributed systems requires correlation across multiple data sources. Metrics, logs, and traces provide complementary views of system behavior. Effective monitoring strategies combine these telemetry types to identify security incidents [10]. Compliance automation utilizes audit logs as a means of showing uninterrupted compliance with security frameworks. SOC 2, HIPAA, and ISO 27001 are security frameworks that require documented proof of the implemented security controls. Automated evidence collection pulls out the most relevant log entries that show the enforcement of encryption, the lifecycle management of credentials, and the effectiveness of access control. Manual effort in compliance reporting decreases significantly through automation.

Observability frameworks must scale with infrastructure growth. Traditional monitoring approaches fail when systems span multiple cloud platforms and generate massive telemetry volumes [10]. Security information and event management systems aggregate logs from distributed sources. Correlation rules identify patterns indicating potential security incidents. Real-time analysis enables rapid response to detected threats. Observability-driven security combines proactive monitoring with automated response capabilities [10].

Table 4. Zero Trust and Compliance Framework [9, 10]

Security Dimension	Implementation Feature
Continuous Verification	Every credential request undergoes authentication and authorization evaluation without implicit network-based trust
Least-privilege Access	Time-limited credentials enforce periodic re-verification, requiring services to demonstrate ongoing authorization
Microsegmentation	Role-specific credential generation provides each service instance with credentials scoped exclusively to operational requirements
Lateral Movement Prevention	Compromised credentials grant access only to explicitly authorized resources within limited time windows
Comprehensive Logging	Captures all encryption operations, credential issuance events, and secret access patterns for audit trail creation
Anomaly Detection	Log aggregation enables security operations teams to identify unusual access patterns and potential credential misuse
Automated Alerting	Triggers incident response workflows when credential issuance deviates from baselines or when expired credentials generate access attempts
Compliance Automation	Extracts audit log entries demonstrating encryption enforcement, credential lifecycle management, and access control effectiveness

Conclusion

Modern cloud infrastructure security demands comprehensive solutions addressing both configuration protection and runtime access control. The integrated framework presented demonstrates how automated YAML encryption combines with ephemeral credential generation to eliminate persistent security vulnerabilities inherent in traditional secrets management. Hierarchical encryption architectures provide cryptographic protection for sensitive configuration parameters while maintaining operational transparency through selective encryption strategies. Asymmetric cryptography enables the separation of encryption and decryption authorities, preventing credential exposure during collaborative development processes. Validation pipelines maintain consistent encryption coverage for all deployment environments and are capable of detecting misconfigurations before a production release. Ephemeral credential systems are at the core of the way access control is revolutionized by the removal of long-lived static credentials and the replacement with dynamically generated, time-limited tokens. A vault-based secret engine is the one that issues credentials that are scoped to particular operational requirements, and these credentials automatically expire

when the task is completed. Role generators are the ones that implement least-privilege access patterns where each service instance is given only the necessary permissions for the intended operations. Linking with continuous deployment workflows is the way through which security controls are ensured to be consistently applied throughout software delivery lifecycles. Cross-platform orchestration patterns make it possible for unified credential management to be carried out across heterogeneous cloud environments, even though the authentication mechanisms may be different. Zero Trust principles become operationalized through continuous verification requirements and microsegmentation capabilities emerging from granular credential scoping. Observability integration provides comprehensive audit trails capturing all encryption operations and credential lifecycle events. Automated compliance evidence collection reduces manual reporting burden while demonstrating continuous adherence to regulatory frameworks. The architectural patterns establish templates for organizations seeking to enhance Infrastructure as Code security without sacrificing operational agility. Future enhancements may incorporate machine learning techniques for anomalous access detection and post-quantum cryptographic algorithms for long-term configuration protection. The demonstrated integration principles offer scalable approaches to credential management, adapting to evolving threat landscapes while supporting multi-cloud deployment strategies. Organizations implementing combined encryption and ephemeral credential frameworks achieve measurable security improvements through reduced attack surfaces and minimized credential exposure windows.

References

[1] Gabriel Arquelau Pimenta Rodrigues et al., "Understanding Data Breach from a Global Perspective: Incident Visualization and Data Protection Law Review," MDPI, 2023. [Online]. Available: https://www.mdpi.com/2306-5729/9/2/27

[2] Yuanhang He et al., "A Survey on Zero Trust Architecture: Challenges and Future Trends," Wiley, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/6476274

[3] Alexander W. Dent, "Fundamental problems in provable security and cryptography," Article submitted to the Royal Society. [Online]. Available: https://eprint.iacr.org/2006/278.pdf

[4] Milan Chauhan and Stavros Shiaeles, "An Analysis of Cloud Security Frameworks, Problems and Proposed Solutions," MDPI, 2023. [Online]. Available: https://www.mdpi.com/2673-8732/3/3/18

[5] Prakash Somasundaram, "UNIFIED SECRET MANAGEMENT ACROSS CLOUD PLATFORMS: A STRATEGY FOR SECURE CREDENTIAL STORAGE AND ACCESS," International Journal of Computer Engineering and Technology (IJCET), 2024. [Online]. Available: https://www.researchgate.net/profile/Prakash-Somasundaram/publication/379435761

[6] I. Indu et al., "Identity and access management in cloud environment: Mechanisms and challenges," ScienceDirect, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098617316750

[7] Ziyue Pan et al., "Ambush from All Sides: Understanding Security Threats in Open-Source Software CI/CD Pipelines," IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, 2022. [Online]. Available: https://arxiv.org/pdf/2401.17606

[8] Yuanbo Li et al., "An Optimal Active Defensive Security Framework for the Container-Based Cloud with Deep Reinforcement Learning," MDPI, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/7/1598