Journal of Science, Technology and Social Transformation Volume 1, Issue 04, 2025

Machine Identities as the New Attack Surface: Securing API Keys, Service Accounts, Workloads, and Autonomous AI Agents in Zero Trust Environments

Santosh Kumar Jadala

Cyber Security & Business Analysis Specialist Independent Researcher USA

Abstract

The rapid expansion of cloud computing, application programming interfaces, microservices, automated workloads, and autonomous AI agents has increased the number and complexity of machine identities operating within enterprise environments. API keys, service accounts, workload credentials, certificates, access tokens, and agent identities frequently possess privileged access to sensitive systems and data, making them attractive targets for credential theft, impersonation, privilege escalation, and lateral movement. Traditional identity and access management approaches remain largely centered on human users and are often inadequate for the scale, speed, and autonomy associated with machine-to-machine interactions. This study examines machine identities as an emerging cybersecurity attack surface and analyzes the security risks associated with API keys, service accounts, cloud workloads, CI/CD identities, and autonomous AI agents. It further evaluates the application of Zero Trust principles to machine authentication, authorization, credential management, continuous verification, and runtime monitoring. Based on threat modeling and security control analysis, the study proposes a Machine Identity Zero Trust Security Framework that integrates identity discovery, credential protection, strong authentication, least-privilege access, contextual authorization, continuous monitoring, automated revocation, and lifecycle governance. The framework provides a structured approach for reducing credential exposure, limiting privilege abuse, and improving accountability across modern enterprise and agent-based systems.

Keywords: Machine Identity; Non-Human Identity; Zero Trust Security; API Security; Service Accounts; Workload Identity;

Introduction

Background and Evolution of Digital Identity

Digital identity has become a central element of enterprise cybersecurity because access to applications, data, infrastructure, and services increasingly depends on the ability to establish who or what is requesting access. Earlier identity and access management systems were designed primarily around human users, including employees, administrators, contractors, customers, and business partners. Authentication commonly depended on usernames and passwords, while authorization decisions were associated with organizational roles, responsibilities, or assigned privileges. Although this model remains important, it no longer represents the complete identity landscape of modern enterprise environments.

Cloud computing, microservices, application programming interfaces, containerized applications, DevOps automation, and distributed systems have significantly increased the number of non-human entities that must authenticate and communicate with other systems. These entities include software applications, service accounts, scripts, virtual machines, containers, certificates, APIs, cloud workloads, deployment pipelines, and automated agents. In many enterprise environments, such identities perform transactions continuously and at a scale far beyond human activity.

This shift has created a need to reconsider how trust is established. Zero Trust security rejects the assumption that an entity should be trusted merely because it operates within an organizational network or has previously been authenticated. Instead, access should depend on explicit verification, least-privilege authorization, and continuous assessment of contextual information (Rose et al., 2020; Syed et al., 2022). These principles are particularly relevant

Corresponding Author: Santosh Kumar Jadala, Cyber Security & Business Analysis Specialist Independent Researcher USA.

How to cite this article: Jadala, S.K. (2025). Machine Identities as the New Attack Surface: Securing API Keys, Service Accounts, Workloads, and Autonomous AI Agents in Zero Trust Environments *Journal of Science, Technology and Social Transformation* 2(1), 1-27.

Source of support: Nil

Conflict of interest: None

to machine identities because automated entities frequently interact across cloud platforms, APIs, and distributed infrastructure where conventional network boundaries offer limited assurance.

Rise of Machine Identities in Modern Enterprises

Machine identities are digital identities assigned to non-human entities to support authentication, authorization, secure communication, encryption, or automated execution. They may include API keys, access tokens, cryptographic certificates, OAuth credentials, service accounts, cloud roles, Kubernetes service accounts, application principals, and workload credentials. Their growth reflects the increasing dependence of organizations on automated and service-oriented computing environments.

For example, a microservice may require credentials to communicate with another service, while a CI/CD pipeline may need privileged access to cloud infrastructure to deploy an application. A Kubernetes workload may require permission to access databases or storage resources. Similarly, autonomous AI agents may interact with APIs, databases, files, communication platforms, and enterprise applications while carrying out assigned tasks.

The security implications of this expansion are substantial. Unlike human identities, which can often be strengthened through multifactor authentication and interactive verification, machine credentials are frequently stored in scripts, application configurations, environment variables, source-code repositories, containers, and deployment systems. Research has demonstrated that sensitive credentials and secret keys can be unintentionally exposed through source-code repositories and software development environments (Meli et al., 2019; Sinha et al., 2015). Similar risks have also been observed in container images and continuous integration workflows (Dahlmanns et al., 2023; Koishybayev et al., 2022).

Consequently, modern organizations must manage machine identities throughout their entire lifecycle, including discovery, registration, credential issuance, authorization, monitoring, rotation, revocation, and eventual decommissioning.

Machine Identities as an Expanding Attack Surface

Machine identities become a significant attack surface when adversaries can steal, misuse, impersonate, or maintain persistent access through their associated credentials. A compromised API key may provide direct access to an external service, while a stolen service-account credential may enable unauthorized access to databases, applications, or administrative systems. Similarly, a compromised workload identity may facilitate lateral movement across cloud services, and exposed CI/CD credentials may provide attackers with access to software deployment infrastructure.

Secret leakage represents one of the clearest examples of this problem. Meli et al. (2019) documented the widespread exposure of authentication secrets in public GitHub repositories, demonstrating the ease with which sensitive credentials can unintentionally become part of source-code histories. Sinha et al. (2015) similarly examined secret-key leakage and mechanisms for detecting exposed credentials within repositories. More recent research has shown that such exposure extends beyond traditional source code. Dahlmanns et al. (2023) identified secrets within container images, while Koishybayev et al. (2022) demonstrated significant security weaknesses within GitHub CI workflows.

The consequences of compromised machine credentials can extend well beyond the initial system. A stolen identity may allow an attacker to impersonate a legitimate service, elevate privileges, access additional resources, establish persistence, or move laterally across interconnected systems. Long-lived credentials increase this risk because a compromised secret may remain valid for extended periods when effective rotation and revocation controls are absent.

Autonomous AI Agents and the Changing Identity Problem

Autonomous AI agents introduce a further complication because they may operate as active participants within enterprise systems rather than merely functioning as passive software components. An agent may interpret instructions, select available tools, call APIs, retrieve information, communicate with other agents, and execute actions on behalf of a user or organization. Security must therefore consider not only whether an agent possesses a valid identity, but also whether its delegated authority is appropriate for each action it attempts to perform.

Delegated access may involve several interconnected stages. A human user may authorize an agent, the agent may invoke a tool, the tool may access an API, and the API may subsequently interact with sensitive enterprise resources. Each stage creates opportunities for excessive privileges, unauthorized delegation, credential misuse, or policy circumvention.

Prompt injection research illustrates the seriousness of this challenge. Greshake et al. (2023) demonstrated that malicious instructions embedded within external content can manipulate language-model-integrated applications. Ruan et al. (2024) further examined risks associated with language-model agents operating within tool-enabled environments, while Zhan et al. (2024) demonstrated the effectiveness of indirect prompt injection against tool-integrated agents.

These findings have significant identity-security implications because an attacker may not need to steal an agent's credentials directly. Instead, an attacker may manipulate an authenticated agent into using legitimate permissions for unauthorized purposes. Autonomous agents therefore require controls such as bounded delegation, tool-specific authorization, contextual access policies,

continuous behavioral monitoring, transaction restrictions, and mechanisms for human intervention.

Research Problem

Current enterprise identity-security models remain heavily influenced by approaches developed for human users and relatively predictable application accounts. These models are increasingly difficult to apply to environments containing large numbers of API credentials, service accounts, cloud workloads, containers, certificates, CI/CD pipelines, and autonomous agents.

Machine identities may be created automatically, operate without direct human supervision, exist for extremely short or long periods, possess excessive privileges, or lack clearly assigned ownership. Some may remain active even after the applications or workloads that originally required them have been removed. Others rely on static credentials that can be copied, exposed, shared, or reused.

Autonomous AI agents further complicate this problem because identity security must account for software entities capable of selecting and executing actions independently within predefined boundaries. An authenticated identity may therefore still present a significant security risk if it possesses excessive authority or if its actions are manipulated through malicious external input. The central research problem addressed in this study concerns how organizations can securely govern diverse machine identities while preserving the automated interactions required by modern enterprise systems.

Research Gap

Existing literature provides substantial guidance on Zero Trust architecture, cloud-native access control, secrets management, OAuth security, container security, and workload protection. However, these areas are often examined independently. Zero Trust research has largely concentrated on architectural principles, access control, continuous verification, and organizational migration strategies (Phiayura & Teerakanok, 2023; Rose et al., 2020; Syed et al., 2022). Software security studies have investigated secret exposure, CI/CD vulnerabilities, and credential leakage in containers (Dahlmanns et al., 2023; Koishybayev et al., 2022; Meli et al., 2019). More recent studies concerning autonomous agents have focused on prompt injection, agent manipulation, and tool misuse (Greshake et al., 2023; Ruan et al., 2024; Zhan et al., 2024).

A significant gap remains in integrating these areas within a unified machine identity security model. Limited research has treated API keys, service accounts, workload identities, certificates, DevOps identities, and autonomous agents as components of the same enterprise identity attack surface. There is also a need to extend Zero Trust principles toward continuous authorization, machine credential lifecycle management, delegated authority, runtime behavior, and identity governance for autonomous systems.

Research Aim and Objectives

The aim of this study is to develop a Zero Trust-oriented security framework for protecting and governing machine identities across modern enterprise environments.

The study seeks to

- classify the major categories of machine identities and their credential mechanisms;
- identify the principal attack surfaces associated with API keys, service accounts, cloud workloads, and autonomous AI agents;
- examine how credential theft, impersonation, privilege escalation, and lateral movement affect machine identities;
- assess the security implications of delegated authority and tool access within autonomous agents;
- evaluate the applicability of Zero Trust principles to machine-to-machine and agent-to-system interactions; and
- propose an integrated framework for machine identity discovery, authentication, authorization, monitoring, credential rotation, revocation, and lifecycle governance.

Research Questions

- RQ1: What security risks distinguish machine identities from conventional human identities?
- RQ2: How do API keys, service accounts, workload identities, and autonomous AI agents expand the enterprise attack surface?
- RQ3: What attack techniques can adversaries use to compromise or abuse machine identities?
- RQ4: How can Zero Trust principles be applied to machine-to-machine and agent-to-system interactions?
- RQ5: What technical and governance controls are required throughout the machine identity lifecycle?

Literature Review and Theoretical Foundations

From Human Identity to Machine Identity

Identity and access management has traditionally focused on authenticating human users and assigning privileges according to organizational roles, responsibilities, and attributes. The expansion of cloud services, automated infrastructure, and machine-to-machine communication has altered this model. Applications, workloads, software services, and autonomous systems increasingly require independent identities to communicate with enterprise resources.

Machine identities differ from human identities in several important respects. They may authenticate thousands of times within short periods, operate continuously, and exist without direct human interaction. Some are temporary and are created dynamically as cloud workloads scale, while others remain active for years. Teng and Rasmussen

(2023) examined the challenges of machine-to-machine authentication and proposed dynamic identity mechanisms intended to reduce dependence on static credentials.

These characteristics make human-centered authentication models insufficient when applied directly to machine environments. Machine identity management requires methods that support automated authentication while maintaining individual identity, controlled authorization, traceability, and accountability.

Machine Identity Management

Machine identity management covers the complete lifecycle of a non-human identity, beginning with discovery and registration and continuing through credential issuance, privilege assignment, authentication, monitoring, rotation, revocation, and decommissioning.

One of the central problems is visibility. Organizations may operate large numbers of machine identities without maintaining a complete inventory of their owners, privileges, dependencies, or operational purpose. Machine accounts may also remain active after the applications or workloads associated with them have been retired, creating orphaned identities that can become unnoticed access paths.

Zero Trust provides a useful foundation for addressing this challenge because it removes assumptions of trust based on network location or organizational ownership. Rose et al. (2020) emphasize that trust should not be implicitly assigned to an asset or account based solely on its physical or network location. Chandramouli and Butcher (2023) extend this approach to cloud-native applications and distributed environments, emphasizing identity-driven policy enforcement across applications and services.

API Keys, Secrets, Tokens, and Credentials

API keys, secrets, and access tokens are widely used because they provide convenient methods for authenticating applications and services. Their simplicity, however, can become a serious weakness when they operate as static bearer credentials. An attacker who obtains a valid credential may be able to use it until it expires or is revoked.

Software repositories remain an important source of credential exposure. Meli et al. (2019) documented substantial secret leakage in public GitHub repositories. Basak et al. (2023) later investigated the practical challenges developers face when dealing with checked-in secrets, demonstrating that credential leakage involves both technical and organizational factors. Rahman et al. (2022) similarly found that secret-detection technologies alone cannot address the entire problem because detection accuracy, remediation procedures, developer practices, and workflow integration influence their effectiveness.

OAuth-based delegated authorization can reduce reliance on simple API keys, but improper implementation can introduce additional vulnerabilities. Fett et al. (2016) conducted a formal security analysis of OAuth 2.0 and identified security properties necessary for secure deployment. Philippaerts et al. (2022) examined security compliance across the OAuth ecosystem and demonstrated the importance of correctly implementing authorization protocols. These findings show that the security of machine credentials depends not only on protecting secrets but also on controlling their scope, duration, issuance, usage, and revocation.

Service Accounts and Application Identities

Service accounts are commonly used by applications, scripts, databases, automation platforms, and background processes. Because these accounts frequently perform operationally important functions, they may receive broad privileges that remain unchanged for long periods.

Such configurations create several risks. Service accounts may depend on long-lived credentials, be shared among multiple applications, lack clearly assigned ownership, or accumulate unnecessary permissions. If compromised, they can provide attackers with persistent access while avoiding controls primarily designed to detect suspicious human logins.

Zero Trust principles provide a basis for addressing these weaknesses. Continuous verification and least privilege are widely recognized as core principles of Zero Trust security (Syed et al., 2022). Applied to service accounts, these principles require organizations to minimize standing privileges, reduce credential lifetime, assign clear ownership, verify workload context, and continuously evaluate whether requested access remains justified.

Cloud-Native Workload Identities

Cloud-native environments create and remove workloads at a scale that traditional enterprise IAM systems were not originally designed to manage. Containers, Kubernetes pods, virtual machines, microservices, and serverless functions may each require identities to communicate with other services and access enterprise resources.

Kubernetes demonstrates this challenge clearly. Workloads may use service accounts and associated tokens to authenticate with resources within the cluster. Poorly configured role-based access control can therefore provide workloads with unnecessary or highly privileged access. Rostami (2023) discusses the importance of RBAC authorization within Kubernetes, while Shamim et al. (2020) identify numerous security practices required to protect Kubernetes infrastructure.

Credential lifetime and identity binding are particularly important in cloud-native environments. A static secret stored inside a container may be copied and reused beyond its intended workload. By contrast, short-lived credentials tied to a specific workload or execution context can limit the usefulness of stolen authentication material. This approach aligns with Zero Trust by requiring authorization decisions to

consider identity and operational context rather than relying solely on possession of a credential.

Zero Trust Security Foundations

Zero Trust constitutes the principal theoretical foundation of this study. Rose et al. (2020) describe Zero Trust as an architectural model that removes implicit trust based on network location and requires authentication and authorization before access to organizational resources is permitted. Central principles include explicit verification, least-privilege access, resource-focused protection, and continuous evaluation.

Subsequent research has examined the practical implementation and limitations of this model. Teerakanok et al. (2021) identify challenges associated with organizational migration toward Zero Trust, while Phiayura and Teerakanok (2023) propose a structured framework for managing the transition. Fernandez and Brazhuk (2024) critically examine Zero Trust Architecture and highlight issues related to identity, policy management, implementation complexity, and operational requirements.

For machine identities, Zero Trust is particularly applicable because service-to-service interactions commonly occur across dynamic and distributed infrastructure. Network position is therefore an unreliable indicator of legitimacy. Access decisions must instead consider verified workload identity, requested resources, transaction context, risk conditions, and the minimum privileges required for the task.

Workload Identity and Zero Trust

Traditional network security often assumes that systems operating within trusted network segments can communicate with fewer restrictions. Cloud-native environments weaken this assumption because workloads may operate across multiple clouds, data centers, containers, temporary infrastructure, and third-party services.

A Zero Trust workload model requires each service or workload to authenticate independently before gaining access to a protected resource. Chandramouli and Butcher (2023) emphasize identity-based access control for cloud-native applications operating across multiple environments. Such an approach can restrict lateral movement because compromise of one service does not automatically provide trusted access to neighboring workloads.

This model also changes the meaning of authentication. Successful authentication should establish the identity of the requesting workload, but it should not automatically provide unrestricted authority. Authorization must still determine whether the workload is permitted to access the requested resource under the current conditions.

Identity Threat Detection and Response

Preventive controls cannot guarantee that machine identities will never be compromised. Organizations therefore require mechanisms for detecting abnormal identity behavior after successful authentication. Identity threat detection and response focuses on identifying suspicious authentication patterns, abnormal resource access, unusual privilege changes, unexpected token usage, and other indicators of compromised credentials.

Behavioral monitoring is particularly suitable for many machine identities because automated workloads often follow predictable activity patterns. A service account that suddenly connects from an unfamiliar environment, accesses previously unused resources, or attempts operations outside its normal functional role may indicate compromise.

This approach aligns with continuous verification within Zero Trust. Authentication should not be considered a permanent trust decision. Instead, access can be reassessed according to behavior, context, resource sensitivity, and changing risk conditions (Kang et al., 2023; Syed et al., 2022).

Autonomous AI Agents as Emerging Digital Identities

Autonomous AI agents extend the concept of machine identity because they can select actions within the permissions granted to them rather than simply executing a fixed sequence of predefined operations. This capability creates an important distinction between authentication and authorization. An agent may be legitimately authenticated while still carrying out an inappropriate action because it has been manipulated, received malicious external instructions, or possesses broader authority than necessary.

Greshake et al. (2023) demonstrated that indirect prompt injection can manipulate applications that integrate language models with external content and tools. Ruan et al. (2024) investigated the risks of language-model agents within simulated environments and showed that the ability to interact with external systems creates practical security concerns. Zhan et al. (2024) further demonstrated indirect prompt-injection attacks against tool-integrated agents.

These studies indicate that protecting an agent's credentials alone is insufficient. Security controls must also restrict what an authenticated agent can do with those credentials. Tool-specific permissions, transaction limits, contextual authorization, behavioral monitoring, human approval requirements, and rapid privilege revocation are therefore necessary components of agent identity security.

Limitations of Existing Approaches

Existing security technologies address individual aspects of machine identity risk, but they are often deployed as separate systems. Identity and access management manages users and applications, privileged access management protects high-value accounts, secret-management platforms store credentials, API gateways control service communication, workload-security systems protect cloud resources, and emerging agent-security controls focus on autonomous systems.

This fragmentation can create gaps in visibility, ownership,

policy enforcement, and incident response. Zero Trust offers a common security foundation, but applying it consistently across different classes of machine identity remains challenging. Current approaches also provide limited guidance for autonomous agents whose permissions may be delegated dynamically and whose actions may change according to external data and instructions.

A unified security approach is therefore required. API keys, service accounts, workload identities, certificates, CI/CD credentials, and autonomous agents should be treated as parts of a common machine identity ecosystem rather than isolated security problems. Such an approach should combine identity discovery, credential protection, strong authentication, least-privilege authorization, continuous verification, behavioral monitoring, automated credential revocation, and lifecycle governance. This requirement provides the foundation for the threat model and Machine Identity Zero Trust Security Framework developed in the subsequent sections.

Research Methodology

Research Design

This study adopts a conceptual security framework development approach supported by structured literature analysis, threat modeling, risk assessment, and control mapping. The methodology is designed to examine machine identities as a common security problem across cloud platforms, application environments, DevOps pipelines, workload infrastructures, and autonomous agent systems. Rather than treating API keys, service accounts, workload identities, certificates, and autonomous AI agents as isolated security concerns, the study evaluates them as interconnected forms of non-human identity that require consistent authentication, authorization, monitoring, and lifecycle governance.

The research design combines descriptive and analytical methods. The descriptive component identifies the major forms of machine identity currently used in enterprise environments and documents the credentials, privileges, operational roles, and security weaknesses associated with each category. The analytical component evaluates how these identities can be compromised or abused and maps the identified risks to Zero Trust principles, including explicit verification, least-privilege access, continuous authorization, contextual policy enforcement, and continuous monitoring. This approach is consistent with the resource-centered and identity-driven principles established within Zero Trust architecture (Rose et al., 2020; Syed et al., 2022).

The methodology does not assume that all machine identities present the same level of risk. A short-lived workload token used by a single container, for example, creates a different exposure profile from a long-lived administrative service account or an autonomous agent capable of accessing several enterprise tools. The analysis therefore considers differences in credential lifetime, privilege, autonomy, connectivity, resource sensitivity, and potential compromise impact.

Literature and Standards Analysis

The first stage of the research consists of a structured analysis of academic literature and recognized technical standards related to machine identity security. The review covers Zero Trust architecture, identity and access management, workload authentication, cloud-native access control, API credential security, secrets management, CI/CD security, container security, OAuth authorization, and autonomous agent security.

Zero Trust publications provide the main architectural foundation for the study. NIST SP 800-207 establishes that trust should not be granted automatically on the basis of network location or resource ownership and that authentication and authorization should occur before access to protected resources is permitted (Rose et al., 2020). NIST SP 800-207A extends these principles to cloud-native and multi-location environments and emphasizes identity-based access control for applications and services (Chandramouli & Butcher, 2023). These principles are used as the reference point for evaluating whether existing machine identity practices provide adequate authentication, authorization, and lifecycle controls.

The literature analysis also considers empirical research on credential exposure and software supply-chain security. Studies of public source-code repositories have demonstrated that API keys, passwords, and other authentication secrets are frequently committed to repositories, creating opportunities for unauthorized access (Meli et al., 2019; Sinha et al., 2015). Similar risks have been documented in container images and continuous integration workflows, showing that credential exposure can persist beyond application source code and enter deployment infrastructure (Dahlmanns et al., 2023; Koishybayev et al., 2022).

Research on autonomous agents is also included because these systems introduce a different form of machine identity risk. Tool-enabled agents may possess legitimate credentials while still performing unauthorized actions if their decision process is manipulated. Studies of indirect prompt injection and agent security demonstrate that malicious external instructions can influence agents that interact with tools and external resources (Greshake et al., 2023; Ruan et al., 2024; Zhan et al., 2024).

The reviewed literature is therefore used not only to identify known vulnerabilities but also to establish the security requirements used in the proposed framework.

Machine Identity Classification Method

A taxonomy is developed to classify machine identities according to their operational characteristics and associated security properties. Each identity type is evaluated using seven dimensions:

- identity type;
- credential mechanism;
- primary operational function;
- privilege level;
- credential persistence;
- degree of autonomy; and
- accessible resources.

This classification distinguishes between identities that are manually provisioned and those generated dynamically by cloud platforms or orchestration systems. It also separates static credentials from short-lived and context-bound identities.

API keys and service-account passwords, for example, may remain valid for extended periods unless manually rotated. Workload identities can instead be issued dynamically for a specific application or runtime context. Autonomous agents present a further distinction because their risk depends not only on the credential itself but also on the agent's ability to select tools and initiate actions.

Machine-to-machine identity research supports the need for more dynamic identity models that reduce dependence on persistent shared secrets (Teng & Rasmussen, 2023). The classification developed in this study therefore considers credential lifetime and identity binding as important indicators of potential exposure.

Threat Modeling Method

Threat modeling is used to identify how machine identities may be compromised and how a compromised identity may be used against enterprise resources. The analysis begins by identifying protected assets, including API endpoints, application services, databases, cloud infrastructure, source-code repositories, deployment systems, enterprise data, and AI-enabled tools.

The threat model considers external attackers, malicious insiders, compromised developers, supply-chain attackers, compromised applications, and manipulated or rogue autonomous agents. For each machine identity class, the study examines possible attack paths involving:

- credential theft;
- secret leakage;
- identity impersonation;
- credential replay;
- privilege escalation;
- unauthorized resource access;
- lateral movement;
- persistence;
- supply-chain compromise; and
- abuse of delegated authority.

Particular attention is given to the distinction between credential compromise and authorized identity misuse. In conventional attacks, an adversary may steal an API token and directly impersonate the service. In an agent-based system, however, an attacker may manipulate a legitimate agent into using its own permissions in a harmful way. This distinction is important because successful authentication alone cannot determine whether an action is appropriate. Prompt injection research illustrates how an authenticated tool-enabled agent may be induced to perform actions that conflict with the intent of the original user or system policy (Greshake et al., 2023; Zhan et al., 2024).

Risk Assessment

The identified attack scenarios are assessed using a qualitative multidimensional risk model. The assessment considers seven primary variables:

Risk = f(Likelihood, Impact, Privilege, Credential Exposure, Autonomy, Blast Radius, Detection Difficulty)

Likelihood represents the probability that an identity or its credentials can be compromised. Impact measures the consequences of successful misuse. Privilege represents the level of authority available to the identity, while credential exposure reflects how easily authentication material can be discovered, copied, or reused.

Autonomy is particularly important for machine identities capable of making independent decisions. A static API client with access to one endpoint presents a different security profile from an autonomous agent able to select among multiple tools, databases, and cloud services. Blast radius measures how many resources or systems could be affected following compromise, while detection difficulty evaluates how easily malicious activity could resemble legitimate machine behavior.

Each identity is rated across these dimensions using predefined low, medium, high, and critical categories. The purpose is not to assign unsupported empirical probabilities, but to provide a structured basis for comparing identity types and identifying where stronger controls are required.

Framework Development

The proposed Machine Identity Zero Trust Security Framework is developed by mapping the risks identified during threat modeling to appropriate security controls. The framework follows the principle that possession of a valid credential should not result in permanent or unrestricted trust.

- Security controls are organized around seven functions:
- machine identity discovery and inventory;
- credential and secrets protection;
- machine and workload authentication;
- least-privilege and contextual authorization;
- runtime protection for workloads and autonomous agents;
- continuous detection and response; and
- governance and lifecycle management.

The framework also incorporates short-lived credentials, automated rotation, workload-bound authentication, policy-based authorization, identity behavior monitoring, and rapid revocation. These controls reflect the Zero Trust requirement that access be explicitly evaluated and limited according to

current conditions rather than granted indefinitely after an initial authentication event (Rose et al., 2020; Fernandez & Brazhuk, 2024).

Framework Evaluation

The proposed framework is evaluated using representative attack scenarios derived from the machine identity taxonomy. These scenarios include leaked API keys, compromised service-account credentials, stolen Kubernetes workload tokens, exposed CI/CD secrets, cloud identity abuse, and autonomous agent misuse.

For each scenario, the evaluation examines the attack path under a conventional identity model and compares it with the expected outcome when the proposed Zero Trust controls are applied. The analysis focuses on whether the framework can reduce credential usability, restrict excessive permissions, limit lateral movement, detect abnormal behavior, and shorten the period between compromise and revocation.

Evaluation criteria include credential lifetime, privilege scope, identity traceability, authorization granularity, detection capability, revocation speed, and potential blast radius. This scenario-based evaluation allows the proposed framework to be assessed without presenting simulated values as real-world experimental evidence.

Taxonomy of Machine Identities and Enterprise Attack Surface

Modern enterprise environments contain a diverse population of non-human identities that operate across applications, cloud infrastructure, development systems, automation platforms, and AI-enabled services. Although these identities differ in implementation, they share a common security characteristic: each represents a mechanism through which a machine, workload, application, or autonomous system is recognized and authorized to access resources.

The attack surface created by machine identities extends beyond the credentials themselves. Risk also arises from excessive privileges, weak lifecycle management, poor ownership, insufficient monitoring, persistent authorization, and dependencies between services. A leaked API key and an overprivileged workload token may therefore produce similar consequences even though they originate from different technologies.

API Keys and Access Tokens

API keys are among the most common forms of machine authentication. They are frequently used to allow applications, scripts, and external services to access APIs without interactive authentication. Their operational simplicity, however, often results in weak security practices.

Many API keys function as bearer credentials, meaning that access is granted primarily on the basis of possession. If an attacker obtains the key, the service may be unable to distinguish the attacker from the legitimate application unless additional contextual controls are present. Static keys are especially risky when they remain valid for long periods or have broad permissions.

Credential exposure may occur through source-code repositories, configuration files, command histories, logs, environment variables, collaboration platforms, and build systems. Meli et al. (2019) documented extensive secret leakage in public GitHub repositories, while Basak et al. (2023) found that developers continue to face practical difficulties in preventing and remediating checked-in secrets.

Access tokens can reduce some risks when they are short-lived and narrowly scoped. However, token theft and replay remain possible when authorization servers and resource servers do not apply adequate restrictions. OAuth research has shown that secure delegated authorization depends strongly on correct protocol implementation and validation (Fett et al., 2016; Philippaerts et al., 2022).

Service Accounts

Service accounts provide identities for applications, background processes, databases, scheduled tasks, and automation systems. They are essential for machine-to-machine operations but frequently become high-value targets because they may possess broad or privileged access.

A common weakness is the use of long-lived passwords or tokens that are rarely rotated. Service accounts may also be shared across multiple applications, making it difficult to determine which process performed a particular action. Ownership can become unclear when applications are retired, teams are reorganized, or credentials are inherited by new systems.

The main risks include credential theft, privilege escalation, persistent unauthorized access, and lateral movement. If a service account has permissions across several systems, compromise of a single credential can expose a much larger portion of the enterprise environment. Applying least privilege and separating identities according to individual service functions can reduce this risk.

Application and Cloud Identities

Cloud platforms increasingly provide application identities through managed identities, service principals, cloud roles, and temporary security tokens. These mechanisms can reduce dependence on manually stored passwords and API keys because credentials may be generated dynamically by the underlying platform.

The security of cloud identities depends heavily on privilege configuration. Excessive role assignments, permissive trust policies, inherited permissions, or weak cross-account controls can allow compromised workloads to access resources beyond their intended scope.

Cloud-native Zero Trust models emphasize that identity should remain central to authorization regardless of the workload's network location. Chandramouli and Butcher (2023) argue for application and service identity-based

access control across distributed cloud-native environments. This approach is important because workloads often move between hosts, clusters, regions, and cloud platforms, making network location an unreliable indicator of trust.

Workload Identities

Workload identities represent virtual machines, containers, Kubernetes pods, microservices, and serverless applications. These identities are increasingly important as organizations replace monolithic applications with distributed services that communicate continuously.

The principal security challenge is ensuring that credentials remain bound to the workload for which they were issued. Static secrets embedded in container images or deployment manifests may be extracted and reused elsewhere. Dahlmanns et al. (2023) demonstrated that sensitive secrets can be exposed through container images, confirming that deployment artifacts themselves can become sources of credential leakage.

Kubernetes environments present similar risks. Service accounts are commonly used to provide pods with identities for interacting with cluster resources. Improperly configured RBAC policies can grant unnecessary privileges, which may enable privilege escalation or cluster-wide compromise following token theft (Rostami, 2023). Kubernetes security research therefore emphasizes restrictive authorization, isolation, and careful configuration of workload permissions (Shamim et al., 2020).

Short-lived workload credentials and identity attestation can reduce these risks by ensuring that credentials remain useful only within a limited operational context.

Certificates and Cryptographic Machine Credentials

Digital certificates provide machine identities for encrypted communication, mutual authentication, service-to-service connections, and infrastructure security. They are widely used in web servers, service meshes, virtual private networks, Internet of Things devices, and internal enterprise systems.

The security of certificate-based identities depends on protection of the associated private keys and effective certificate lifecycle management. A stolen private key may allow an attacker to impersonate a legitimate server or service even when the certificate itself remains valid.

Operational weaknesses include expired certificates, undocumented certificates, weak key storage, duplicate credentials, and failure to revoke compromised certificates. Large-scale environments may contain thousands of certificates distributed across applications and infrastructure, making manual management unreliable.

Automated issuance, rotation, renewal, inventory, and revocation are therefore central to certificate-based machine identity security.

CI/CD and DevOps Identities

CI/CD systems require substantial machine privileges because they interact with source-code repositories, package registries, cloud platforms, artifact stores, infrastructure-as-code systems, and production deployment environments. This makes pipeline identities particularly attractive targets.

Credentials used by CI/CD systems may include repository tokens, deployment keys, cloud credentials, signing keys, and automation secrets. If compromised, these credentials can allow attackers to manipulate software before it reaches production or gain direct access to deployment infrastructure.

Koishybayev et al. (2022) identified security concerns within GitHub CI workflows, demonstrating that weaknesses in workflow configuration can expose organizations to significant supply-chain risks. The danger is amplified when a single pipeline credential has access to multiple repositories, environments, or production systems.

Zero Trust controls for CI/CD identities should therefore include short-lived deployment credentials, environment-specific permissions, separation of duties, provenance verification, protected execution environments, and continuous monitoring of pipeline behavior.

Autonomous AI Agent Identities

Autonomous AI agents represent an emerging class of machine identity because they can make decisions, select tools, invoke APIs, and carry out sequences of actions within delegated authority. Their security profile differs from traditional service accounts because the behavior of an agent may change according to instructions, external content, context, and interactions with other systems.

Agent credentials may provide access to email services, databases, cloud platforms, browsers, enterprise software, external APIs, or code execution tools. A highly privileged agent can therefore become a powerful attack path if its instructions are manipulated.

Greshake et al. (2023) demonstrated that indirect prompt injection can influence language-model-integrated applications through malicious external content. Zhan et al. (2024) extended this concern to tool-integrated agents, showing that adversarial instructions can affect agents that interact with external systems. Ruan et al. (2024) likewise identified security risks associated with language-model agents operating in environments where their decisions can produce external effects.

The main risks include excessive delegated authority, unauthorized tool use, data disclosure, credential misuse, unsafe transactions, confused deputy behavior, and manipulation through untrusted input. Securing agent identities therefore requires more than authentication. Organizations must also enforce action-level authorization, tool restrictions, context-aware policies, transaction boundaries, continuous monitoring, and human approval for high-impact operations.

Table 1: Taxonomy of Machine Identities, Credentials, and Security Risks

Machine Identity	*Typical Credential*	*Primary Function*	*Major Security Risks*	*Zero Trust Requirement*
API/Application Identity	API key, access token, OAuth token	Access to APIs and application services	Credential leakage, replay, excessive scope, unauthorized API use	Short-lived and narrowly scoped credentials with contextual verification
Service Account	Password, token, certificate	Background processes and automated system operations	Excessive privilege, persistent credentials, shared accounts, lateral movement	Unique identity, least privilege, rotation, continuous monitoring
Cloud Application Identity	Cloud role, service principal, temporary token	Access to cloud resources and services	Role misconfiguration, privilege escalation, cross-account abuse	Context-aware authorization and temporary privileges
Container/ Kubernetes Workload	Service-account token, workload certificate	Service-to-service and cluster access	Token theft, excessive RBAC privileges, workload impersonation	Workload-bound identity, short-lived tokens, restrictive RBAC
Virtual Machine/ Serverless Workload	Instance role, managed identity, temporary token	Access to cloud applications, storage, and databases	Metadata credential theft, privilege abuse, workload compromise	Identity attestation and resource-specific authorization
Certificate Identity	Certificate and private key	Machine authentication and encrypted communication	Private-key theft, certificate misuse, unmanaged expiration	Automated issuance, rotation, inventory, and revocation
CI/CD Pipeline Identity	Deployment token, repository key, cloud credential	Build, testing, release, and deployment automation	Supply-chain compromise, secret leakage, unauthorized deployment	Ephemeral credentials and environment-specific access
Autonomous AI Agent	API token, delegated credential, agent identity	Tool use, API access, decision execution, automated tasks	Tool misuse, confused deputy attacks, excessive autonomy, unauthorized actions	Bounded delegation, action-level authorization, continuous monitoring, human escalation
Shadow/ Orphaned Identity	Any persistent credential	Unknown, outdated, or unmanaged function	Unmonitored access, credential persistence, unknown ownership	Continuous discovery, ownership verification, recertification, automatic decommissioning

Shadow and Orphaned Machine Identities

Shadow and orphaned identities represent machine accounts or credentials that exist outside formal inventory and governance processes. Shadow identities may be created by development teams, applications, contractors, or automated systems without centralized registration. Orphaned identities remain active even though the original application, workload, employee, or business process no longer requires them.

These identities create significant security exposure because they may retain valid permissions without active ownership or monitoring. Attackers who obtain such credentials may operate for long periods before their activity is identified.

The problem is particularly serious in dynamic cloud and DevOps environments where identities can be created rapidly. Effective governance therefore requires continuous identity discovery, ownership assignment, credential-age monitoring, access reviews, and automated decommissioning.

The taxonomy demonstrates that machine identity risk cannot be reduced to credential secrecy alone. The severity of exposure depends on the relationship between the identity, its privileges, its operational environment, and the resources it can reach. Static API keys create risk because possession may be sufficient for authentication, while cloud workloads create risk when identity is weakly bound to runtime context. Service accounts become dangerous when privileges remain excessive, and autonomous agents introduce additional concerns because legitimate authorization can be misused through manipulated decision processes.

This classification provides the basis for the detailed threat model presented in the next section, where the study examines how adversaries can move from credential exposure or identity manipulation to impersonation, privilege escalation, lateral movement, persistence, and unauthorized autonomous execution.

Threat Model for Machine Identity Compromise

Protected Assets

Machine identities provide access to assets that are central to enterprise operations, including application programming interfaces, databases, cloud services, source-

code repositories, container orchestration platforms, CI/CD infrastructure, cryptographic material, business applications, and sensitive organizational data. The value of a machine identity is therefore determined not only by the credential attached to it but also by the resources, permissions, and relationships available to that identity.

In cloud and service-oriented environments, a single machine identity may communicate with several systems. A service account may access a database and an internal API, while a deployment account may interact with source-code repositories, artifact registries, cloud infrastructure, and production environments. Consequently, compromise of one identity can expose resources located far beyond the system where the credential was initially obtained.

From a Zero Trust perspective, resources should be protected individually rather than treated as trusted simply because they exist within an organizational network. Rose et al. (2020) emphasize that access decisions should focus on resources and verified identities rather than assumptions based on network location. This principle is particularly important for machine identities because workloads frequently operate across cloud regions, clusters, platforms, and external services.

The assets considered within the threat model include machine credentials themselves, enterprise data, API endpoints, cloud control planes, workload environments, software repositories, deployment systems, autonomous agent tools, and the authorization policies governing access to these resources.

Threat Actors

Machine identities may be targeted or abused by several categories of threat actors. External attackers may seek exposed API keys, tokens, certificates, or service-account credentials to bypass normal authentication procedures. Malicious insiders may already possess legitimate access to development systems and may exploit machine credentials to obtain privileges beyond those assigned to their personal accounts. Compromised developers and administrators can also provide an indirect path to machine credentials stored in repositories, local environments, or deployment systems.

Supply-chain attackers represent another important category because CI/CD platforms and software delivery systems often contain credentials with substantial access to production environments. Koishybayev et al. (2022) demonstrated that weaknesses in continuous integration workflows can create security exposures that affect the broader software supply chain.

The introduction of autonomous AI agents adds another threat source. An agent does not have to be intentionally malicious to produce harmful behavior. It may be manipulated through adversarial content, compromised instructions, or unauthorized delegation. Greshake et al. (2023) showed that indirect prompt injection can influence language-model-integrated applications through external content. In such cases, the attacker may exploit the authority already assigned to the agent rather than directly compromising its authentication credential.

Credential Theft and Secret Leakage

Credential theft represents one of the most direct paths to machine identity compromise. API keys, passwords, tokens, private keys, and deployment credentials may be exposed through source code, configuration files, environment variables, logs, container images, shared documents, command histories, or CI/CD systems.

Research indicates that secret leakage remains a persistent problem in software development. Meli et al. (2019) documented widespread exposure of authentication secrets in public GitHub repositories. Sinha et al. (2015) similarly examined secret-key leakage in source-code repositories and the need for automated detection and remediation. The problem can continue into later stages of the software lifecycle. Dahlmanns et al. (2023) identified secrets within publicly available container images, indicating that credentials can become embedded within deployment artifacts even when they are absent from current source code.

The severity of credential leakage depends partly on credential lifetime and scope. A short-lived token restricted to one workload and one resource may lose its usefulness quickly. A static API key with broad access may remain exploitable for months if the organization does not detect the exposure.

Credential security must therefore include prevention, detection, rotation, and revocation. Rahman et al. (2022) found that secret-detection tools alone are insufficient because successful remediation also depends on developer practices and organizational processes. Machine identity protection must consequently address both technical controls and lifecycle governance.

Machine Identity Impersonation

Once an attacker possesses valid machine credentials, the compromised identity can often be impersonated. This attack is especially dangerous when authentication relies primarily on possession of a bearer token or static secret.

From the receiving system's perspective, malicious requests may appear indistinguishable from legitimate machine activity. If an attacker uses a valid API key associated with an application, the API may treat the attacker as the authorized application. Similarly, possession of a service-account password or private key may allow an adversary to communicate with other systems under the identity of a legitimate service.

Impersonation weakens accountability because audit records may correctly identify the machine identity while failing to identify the actual actor controlling it. The problem supports the use of context-bound credentials, workload attestation, short credential lifetimes, and behavioral

monitoring rather than relying exclusively on credential validity.

Machine-to-machine authentication research has recognized the limitations of persistent identity credentials. Teng and Rasmussen (2023) propose more dynamic approaches to machine identity, reflecting the broader need to bind authentication to the current machine or operating context.

Privilege Escalation

Machine identity compromise becomes more serious when the affected account has excessive permissions. Service accounts, deployment identities, and cloud workloads are often granted broad privileges for operational convenience. Over time, additional permissions may be added without removing privileges that are no longer required.

Attackers can exploit these privileges directly or use the compromised identity to obtain access to more powerful roles. In Kubernetes, for example, excessive RBAC permissions may enable a compromised service account to access secrets, create privileged workloads, or interfere with cluster resources. Rostami (2023) highlights the importance of careful role-based authorization in Kubernetes environments.

Privilege escalation is not limited to traditional workloads. An autonomous agent with access to several tools may obtain effective privilege through the combination of otherwise separate capabilities. Permission to read files, call an API, and send external messages may create a path for data extraction even when no individual permission appears highly privileged.

Least-privilege authorization should therefore be evaluated in terms of complete machine capabilities and potential access chains rather than individual permissions alone.

Lateral Movement

Machine identities can provide attackers with pathways between interconnected services. After compromising one workload, an adversary may use its credentials to communicate with another internal service, retrieve additional secrets, or access management systems.

Traditional network trust can make this process easier when internal services accept requests based largely on network location. Zero Trust seeks to limit this weakness by requiring identity-based authentication and authorization for each protected resource (Chandramouli & Butcher, 2023).

Machine-to-machine environments are particularly susceptible to lateral movement because application dependencies create legitimate communication paths between services. An attacker who controls one trusted workload may attempt to follow these same paths. Microsegmentation, service-specific authorization, workload identities, and short-lived credentials can restrict the number of systems accessible from a compromised identity.

Persistence Through Machine Credentials

Machine credentials can provide long-term persistence when they remain valid after the initial compromise. Unlike a suspicious human login that may trigger account recovery or password changes, compromised service credentials may remain unnoticed because automated activity continues to resemble expected system behavior.

Long-lived API keys, unmanaged service accounts, forgotten certificates, and orphaned cloud identities are especially suitable for persistence. An attacker may retain a credential even after the original vulnerability used to obtain it has been corrected.

Credential rotation is therefore a security requirement rather than an administrative convenience. Automated expiration and revocation reduce the period during which stolen authentication material remains useful. Continuous machine identity inventory is also necessary to identify credentials that no longer have an active operational purpose.

Supply-Chain and CI/CD Identity Compromise

CI/CD pipelines occupy a sensitive position within the machine identity attack surface because they connect software development with production deployment. Pipeline identities may possess permission to retrieve code, create packages, sign artifacts, modify infrastructure, and deploy applications.

Compromise of these identities can therefore allow an attacker to affect both software and infrastructure. Koishybayev et al. (2022) identified several security weaknesses in GitHub CI workflows, illustrating how workflow configuration can expose secrets and enable unauthorized behavior.

A compromised deployment credential may also bypass controls that focus primarily on human administrators. Since automated deployment activity is expected within CI/CD environments, malicious actions performed through a valid pipeline identity may appear legitimate.

Security controls should include separation between development and production identities, temporary deployment credentials, protected execution environments, restricted repository permissions, provenance verification, and detailed audit records.

Agent Credential and Tool Abuse

Autonomous AI agents create a distinct attack path because valid credentials may be misused without being stolen. An attacker may manipulate an agent into performing an action using the credentials already available to it.

Consider an agent authorized to read organizational documents and send email messages. Malicious instructions embedded in a retrieved document could attempt to persuade the agent to transmit sensitive information externally. The agent remains correctly authenticated throughout the incident, but the intended relationship between identity and authority has been violated.

Greshake et al. (2023) demonstrated the feasibility of indirect prompt injection against systems that process untrusted external content. Zhan et al. (2024) further examined such attacks in tool-integrated agent environments, where manipulated instructions can influence external actions.

Agent security therefore requires separation between instructions, data, credentials, and tool permissions. Authentication establishes which agent is making a request, but authorization must determine whether the specific requested action is appropriate within the agent's assigned purpose.

Machine Identity Attack Chains

Machine identity attacks often involve several stages rather than a single security failure. A representative attack chain may begin with secret exposure in a repository, followed by credential theft, identity impersonation, unauthorized API access, privilege escalation, lateral movement, and eventual access to sensitive information.

A CI/CD attack could follow a similar sequence

Exposed pipeline credential → CI/CD identity impersonation → unauthorized deployment access → malicious application modification → production compromise

An autonomous agent attack may instead follow:

Malicious external content → indirect prompt injection → manipulation of authenticated agent → unauthorized tool call → sensitive resource access → data disclosure or unauthorized action

These examples demonstrate that machine identity security cannot focus solely on preventing credential theft. Effective protection requires controls throughout the entire attack chain, including credential protection, identity verification, least-privilege authorization, runtime restrictions, behavioral monitoring, rapid revocation, and resource isolation.

Autonomous AI Agents as a New Machine Identity Security Challenge

From Passive Applications to Autonomous Actors

Traditional machine identities generally belong to software that performs predetermined functions. A database service account accesses a database, a deployment identity performs software releases, and an API client sends requests according to application logic. Autonomous AI agents differ because they may interpret objectives, choose between actions, select tools, process external information, and determine how to complete assigned tasks.

This difference changes the security significance of machine identity. For conventional software, identity protection is largely concerned with preventing unauthorized parties from gaining control of credentials. For autonomous agents, security must also consider whether the legitimate agent itself is making an authorized decision.

Ruan et al. (2024) examined the risks of language-model agents operating within environments where their choices can lead to external consequences. Their findings support the need to evaluate agents not simply as software models but as operational entities whose decisions can affect real systems.

Agent Authentication

Every autonomous agent that accesses protected resources should possess a unique and verifiable identity. Shared credentials between multiple agents weaken accountability because security teams cannot determine which agent initiated a specific operation.

Authentication should identify the individual agent, its operating environment, and where appropriate, the user or service on whose behalf it is acting. An agent instance should not automatically inherit permanent access simply because another instance of the same application was previously authenticated.

Short-lived credentials are particularly suitable for agent environments because agent instances may be created temporarily to complete specific tasks. Credentials can be issued for the duration of the task and revoked when execution ends.

Such practices reflect the Zero Trust principle that authentication should be based on the current entity and operational context rather than permanent trust relationships (Rose et al., 2020).

Delegated Authority

Agent security is closely connected to delegation. A user or organization may grant an agent permission to perform specific activities on its behalf. This relationship can be represented as:

Human or System → Agent → Tool → API → Enterprise Resource

Security controls must preserve the limits of the original delegation throughout the chain. If a user authorizes an agent to summarize emails, this permission should not automatically allow the agent to delete messages, modify account settings, or send confidential attachments to external recipients.

Delegated authority should therefore specify permitted resources, actions, duration, purpose, and conditions. The receiving system should also be able to determine whether the agent is acting independently or on behalf of another identity.

Agent Tool and API Permissions

Tools convert agent decisions into external actions. An agent may have access to databases, browsers, email services, cloud platforms, enterprise applications, source-code repositories,

or code-execution environments. The security significance of the agent therefore depends heavily on its available tools.

Broad tool permissions can create unnecessary risk. An agent assigned to retrieve information from a database may require read access but not permission to modify or delete records. Similarly, an agent responsible for drafting an email should not automatically receive permission to send messages without approval if the business process requires human review.

Tool authorization should therefore operate independently of the agent's general authentication status. Access should be granted at the level of specific actions, resources, and transactions.

Agent-to-Agent Trust

Multi-agent systems introduce additional identity relationships because agents may exchange information, delegate tasks, or request actions from other agents. Trusting messages merely because they originate from another component within the same system can reproduce the weaknesses of traditional perimeter security.

Each agent should be independently identifiable, and inter-agent requests should be authenticated and authorized according to defined policies. Delegation should also be traceable. If Agent A asks Agent B to perform an action on behalf of a user, audit records should preserve the relationship between the user, Agent A, Agent B, and the final action.

Without such controls, a compromised agent may exploit trusted relationships to influence more privileged agents.

Prompt Injection as an Identity and Authorization Problem

Prompt injection is often treated mainly as an input-security problem. In agent environments, however, it also becomes an authorization problem because manipulated instructions can influence how a legitimate identity uses its permissions.

Greshake et al. (2023) showed that malicious instructions embedded in external information can alter the behavior of language-model-integrated applications. Zhan et al. (2024) demonstrated similar risks in tool-integrated agents.

The important distinction is that authentication may operate correctly during such an attack. The system knows which agent is making the request, and the agent possesses legitimate credentials. The failure occurs because the agent's authority is broad enough to perform an action that should not have followed from untrusted external content.

Authorization systems must therefore examine the requested action independently of the instructions generated by the agent. High-risk operations should require policy validation or human approval even when the request originates from an authenticated agent.

Agent Credential Theft

Autonomous agents can also experience conventional credential compromise. Tokens, API keys, session credentials, and service secrets may be accessible through the agent's execution environment, memory, tool configuration, logs, or connected applications.

Credential material should not be unnecessarily exposed within agent prompts or model-visible context. Where possible, agents should request actions through controlled intermediaries rather than directly possessing long-lived credentials. A policy enforcement layer can retain the credential and issue authorized requests after verifying the agent's identity and requested action.

This arrangement limits the possibility that sensitive secrets are disclosed through model output, malicious instructions, or compromised context.

Confused Deputy Attacks

A confused deputy attack occurs when an entity with legitimate authority is manipulated into using that authority on behalf of an unauthorized party. Autonomous agents are particularly susceptible because they process instructions and external information while simultaneously possessing access to tools.

An attacker who cannot directly access a protected database may attempt to convince a privileged agent to retrieve the information on their behalf. The agent becomes the deputy through which the attacker exercises authority.

Preventing this form of abuse requires purpose-aware authorization. Systems should evaluate not only whether the agent has permission to access the resource but also whether the access corresponds to an authorized task, user, and transaction.

Agent Autonomy and Blast Radius

Autonomy can increase operational value, but it can also enlarge the consequences of security failure. An agent capable of independently reading data, invoking APIs, modifying systems, and communicating externally can produce a greater blast radius than an agent restricted to generating recommendations.

Risk should therefore be assessed according to the combination of autonomy and privilege. Highly autonomous agents with access to sensitive tools require stronger controls than advisory agents operating without external permissions.

Transaction limits, resource restrictions, execution isolation, approval thresholds, and rapid shutdown mechanisms can restrict the maximum impact of unexpected or malicious behavior.

Human Oversight and Agent Identity Governance

Human oversight remains important for actions involving financial transactions, infrastructure changes, sensitive data disclosure, security-policy modifications, or other high-impact operations. Human review should be applied according to risk rather than required for every routine agent interaction.

Agent governance should specify who owns each agent, what tasks it may perform, which tools it may use, what credentials it can obtain, and under what conditions human approval is required. Logs should record the agent identity, delegated user or system identity, requested operation, tool invoked, authorization decision, and resulting action.

These controls provide accountability and allow organizations to suspend or revoke an agent's authority when abnormal behavior is detected.

Zero Trust Security Model for Machine Identities

Extending Zero Trust Beyond Human Users

Zero Trust was developed to remove implicit trust from access decisions and protect resources according to identity, context, and policy. These principles apply directly to machine identities. API clients, workloads, service accounts, deployment systems, and autonomous agents should not receive trust merely because they operate within organizational infrastructure.

Rose et al. (2020) establish that Zero Trust requires authentication and authorization of subjects and devices before access to enterprise resources is established. Chandramouli and Butcher (2023) extend these ideas to cloud-native applications, where application and service identities become central to policy enforcement.

For machine environments, the principle can be expressed simply: every workload or agent should prove its identity, every request should be authorized, and successful authentication should not create permanent trust.

Explicit Machine Authentication

Every machine-to-machine interaction involving protected resources should use a distinct and verifiable identity. Shared secrets that represent several applications or workloads make attribution difficult and increase the consequences of compromise.

Authentication mechanisms should support the scale and short lifetimes of modern workloads. Certificates, managed workload identities, cryptographic attestation, temporary tokens, and platform-issued credentials can provide stronger assurance than manually distributed static passwords.

Authentication should also determine whether the identity is operating from its expected workload or runtime context. Possession of a copied credential should not necessarily be sufficient to reproduce the identity elsewhere.

Short-Lived and Ephemeral Credentials

Persistent credentials increase the time available for attackers to exploit stolen authentication material. Machine identities should therefore use short-lived credentials wherever operationally feasible.

Temporary credentials reduce risk because exposure does not create indefinite access. They also encourage automated issuance and renewal rather than manual credential management.

This principle is particularly useful for containers, serverless applications, CI/CD jobs, and temporary agents. Credentials can be created when the workload starts and invalidated when its task or execution session ends.

Dynamic machine authentication is consistent with the concerns raised by Teng and Rasmussen (2023), who examine alternatives to persistent identity mechanisms in machine-to-machine communication.

Least-Privilege Authorization

Authentication answers which machine is requesting access, while authorization determines what that machine should be allowed to do. These functions must remain distinct.

Service accounts, workloads, and agents should receive only the privileges required for their assigned tasks. Broad permissions granted for convenience create unnecessary attack paths if the identity is compromised.

Least privilege should also account for combinations of capabilities. An agent may have several individually limited permissions that collectively allow a sensitive operation. Authorization reviews should therefore consider the complete set of actions available to an identity.

Syed et al. (2022) identify least privilege as a central feature of Zero Trust architecture, making it particularly relevant to machine identities that frequently operate without direct supervision.

Just-in-Time Access

Some machine identities require privileged access only during particular operations. Maintaining those permissions permanently creates unnecessary standing privilege.

Just-in-time authorization allows elevated access to be granted only when required and removed when the task is complete. A deployment pipeline, for example, may receive production permissions only during an approved release window. An autonomous agent may receive permission to execute a specific high-impact operation only after policy or human approval.

This model reduces the opportunities available to attackers who compromise an identity outside the period when privileged access is required.

Context-Aware Authorization

Machine access decisions should consider contextual information rather than identity alone. Relevant factors may include the requesting workload, execution environment, target resource, requested operation, time, transaction sensitivity, behavior, and current risk indicators.

For autonomous agents, context can also include the user or system that delegated the task, the approved purpose, and the tool being invoked.

Contextual authorization allows the same machine identity to receive different access decisions under different

circumstances. A service account operating from an approved production workload may be permitted to access a database, while the same credential presented from an unexpected environment may be rejected or subjected to additional verification.

Continuous Verification

Zero Trust does not treat initial authentication as permanent evidence of legitimacy. Access should be reassessed when risk conditions change.

Continuous verification for machine identities can include monitoring authentication events, requested resources, privilege use, workload state, behavioral patterns, and execution context. Sudden deviation from expected activity may justify restricting or revoking access.

Kang et al. (2023) emphasize the continuing role of verification within Zero Trust security. For machine identities, this principle is essential because legitimate credentials can be stolen and legitimate agents can be manipulated after successful authentication.

Microsegmentation and Service-to-Service Controls

Microsegmentation limits communication between workloads to explicitly permitted paths. It reduces the assumption that systems sharing a network should automatically communicate with one another.

Service-to-service authorization can restrict each workload to the specific services required for its operation. A compromised application may therefore be prevented from directly accessing unrelated databases or administrative systems.

This control significantly reduces lateral movement. Even if an attacker successfully compromises one machine identity, additional resources still require separate authorization.

Credential Rotation and Automated Revocation

Credential lifecycle management is essential to machine identity security. Credentials should be rotated regularly, replaced after suspected exposure, and revoked when the identity no longer requires access.

Manual processes are unsuitable for environments containing large populations of rapidly changing workloads. Automated issuance, rotation, expiration, and revocation can reduce both administrative errors and periods of credential exposure.

Revocation should also form part of incident response. If behavioral monitoring indicates that a workload or agent may be compromised, security systems should be able to suspend its identity or reduce its permissions without waiting for lengthy manual procedures.

Identity-Based Monitoring

Network monitoring alone cannot provide sufficient visibility into machine identity abuse because malicious activity may occur through valid service connections. Monitoring should therefore be tied directly to identity.

Relevant records include which identity requested access, which resource was targeted, which privileges were exercised, what action was performed, which credentials were used, and whether the request corresponded with established behavioral patterns.

Identity-based monitoring is particularly important for autonomous agents. Audit trails should connect the agent's identity with the original delegation, the tool selected, the authorization decision, and the resulting action. This makes it possible to distinguish normal autonomous behavior from actions that exceed approved boundaries.

Taken together, these controls establish a Zero Trust security model in which machine identities receive no permanent or implicit trust. Authentication establishes identity, authorization limits permitted activity, contextual controls examine the conditions surrounding each request, and continuous monitoring determines whether access should remain valid. This model provides the security foundation for the Machine Identity Zero Trust Security Framework developed in the next section.

Proposed Machine Identity Zero Trust Security Framework

The proposed Machine Identity Zero Trust Security Framework provides a structured security model for managing API keys, service accounts, cloud workloads, containers, CI/CD identities, certificates, and autonomous AI agents within the same control environment. The framework is based on the principle that possession of a valid machine credential should not create permanent trust. Every identity should be known, authenticated, authorized for a defined purpose, monitored during use, and removed when it is no longer required.

This approach responds to a recurring weakness in machine identity security. Organizations often deploy separate controls for secrets, service accounts, APIs, cloud workloads, and privileged access, while the relationships between these identities remain poorly governed. Zero Trust provides a suitable foundation for correcting this fragmentation because it shifts security decisions away from network location and toward identity, resource sensitivity, access context, and explicit policy enforcement (Rose et al., 2020; Syed et al., 2022). Chandramouli and Butcher (2023) further show that identity-centered policy enforcement is particularly relevant to cloud-native applications operating across distributed environments.

The framework is organized into seven connected layers. Each layer addresses a different stage of the machine identity lifecycle, while a Zero Trust policy engine applies consistent security rules across the entire architecture.

Layer 1: Machine Identity Discovery and Inventory

The first layer establishes visibility over all machine identities operating within the enterprise. Security controls cannot protect identities that have not been identified, classified, and assigned to an accountable owner. Discovery should therefore include API keys, service accounts, application principals, cloud roles, workload identities, certificates, CI/CD credentials, automation accounts, and autonomous agents.

Each discovered identity should be recorded with information describing its owner, purpose, associated workload, authentication mechanism, credential lifetime, privileges, accessible resources, and operational status. Relationships between identities should also be mapped. For example, an application may use a service account that accesses an API, which subsequently interacts with a database. Mapping these dependencies helps identify where compromise of one identity may expose other resources.

The inventory should distinguish active, dormant, shadow, and orphaned identities. Dormant and orphaned accounts present particular risk because they may retain valid privileges without active operational oversight. Continuous discovery is therefore preferable to periodic manual inventories, especially in cloud and container environments where identities can be created and removed rapidly.

Layer 2: Credential and Secrets Protection

The second layer protects the credentials associated with machine identities. Secrets should not be embedded in application source code, configuration files, container images, or CI/CD scripts where they can be unintentionally disclosed. Research on public repositories and container environments has shown that credentials can persist in development and deployment artifacts long after their original use (Dahlmanns et al., 2023; Meli et al., 2019).

Sensitive credentials should instead be maintained in controlled secrets-management systems with encryption, access logging, automated rotation, and strict retrieval policies. Static secrets should be replaced with temporary credentials where the underlying platform supports them. Credential scope should also be minimized so that compromise of one token does not provide unnecessary access to unrelated resources.

For certificates and cryptographic identities, the same principle applies to private-key protection. Certificate issuance, renewal, expiration, and revocation should be automated where possible. The objective is to reduce both unauthorized exposure and the period during which compromised authentication material remains useful.

Layer 3: Authentication and Identity Verification

The third layer establishes whether a machine identity is genuine before access is granted. Authentication should verify more than possession of a password, API key, or token. Where feasible, credentials should be bound to a specific workload, runtime, platform, or cryptographic identity.

Cloud workloads can use managed identities, platform-issued tokens, or workload attestation mechanisms that reduce dependence on manually distributed secrets. Kubernetes and service-to-service environments may use workload certificates or other short-lived authentication mechanisms. Autonomous agents should similarly receive unique identities rather than sharing generic application credentials.

A major requirement is traceability. If several workloads share one credential, audit records cannot reliably determine which workload initiated a request. Unique machine identities therefore strengthen both authentication and accountability. Dynamic identity mechanisms are also preferable where machine instances are temporary or frequently replaced (Teng & Rasmussen, 2023).

Layer 4: Authorization and Least Privilege

Authentication establishes identity, but it should not determine the full extent of access. The fourth layer applies least-privilege authorization so that each machine identity receives only the permissions required for its assigned function.

Authorization can be implemented through role-based access control, attribute-based access control, policy-based access control, or combinations of these approaches. The appropriate method depends on the environment, but the underlying requirement is the same: permissions should be specific, limited, and regularly reviewed.

Service accounts should not retain administrative rights simply because they were granted during initial configuration. Workload identities should be restricted to the resources needed for current operations. CI/CD identities should receive environment-specific deployment permissions, while autonomous agents should be authorized at the level of individual tools and actions.

Just-in-time access can further reduce standing privilege. A machine identity requiring elevated authority for a specific operation can receive temporary access and lose that privilege when the operation ends. This follows the least-privilege and continuous-verification principles associated with Zero Trust architecture (Rose et al., 2020; Syed et al., 2022).

Layer 5: Workload and Agent Runtime Protection

The fifth layer controls how authenticated workloads and agents behave during execution. This layer is important because a legitimate identity can still be abused after successful authentication.

For conventional workloads, runtime controls may include execution isolation, workload attestation, service restrictions, network segmentation, and policy enforcement around sensitive operations. For autonomous AI agents,

runtime protection should include tool restrictions, action boundaries, resource limits, transaction thresholds, and separation between trusted instructions and untrusted external content.

The need for such controls is supported by research on tool-enabled language model agents. Greshake et al. (2023) demonstrated that indirect prompt injection can influence systems that process external content, while Zhan et al. (2024) showed similar risks in tool-integrated agents. These attacks demonstrate that the central problem may not be credential theft. A legitimate agent can be manipulated into using valid permissions in an unintended manner.

For this reason, sensitive agent actions should be evaluated independently of the model's internal decision. High-impact requests may require an external policy check or human approval before execution.

Layer 6: Continuous Detection and Response

The sixth layer monitors machine identities after authentication and authorization. This addresses the possibility that credentials may be compromised, workloads may behave unexpectedly, or agents may attempt actions outside established patterns.

Monitoring should focus on identity-related events such as authentication location, accessed resources, privilege use, unusual API requests, unexpected service communication, failed authorization attempts, and deviations from normal workload behavior. For autonomous agents, records should also include tool selection, delegated authority, requested actions, policy decisions, and external destinations.

Continuous verification is an important Zero Trust principle because identity legitimacy can change after initial authentication (Kang et al., 2023; Syed et al., 2022). A previously valid workload may become compromised, and a legitimate agent may begin performing risky actions after processing malicious input.

Response mechanisms should include automated token revocation, session termination, privilege reduction, workload isolation, agent suspension, and escalation to security personnel. Faster response reduces the time during which a compromised identity can be exploited.

Layer 7: Governance, Auditability, and Lifecycle Management

The seventh layer provides organizational control over the complete machine identity lifecycle. Every machine identity should have a documented owner, business purpose, authorized resource scope, credential policy, review period, and decommissioning condition.

Governance should include periodic access reviews, privilege recertification, credential-age monitoring, ownership verification, policy compliance checks, and removal of unused identities. Audit records should preserve sufficient information to reconstruct important machine actions and determine whether access was appropriate.

Autonomous agents require additional governance because accountability may involve several linked identities. Logs should record the user or system that delegated a task, the agent that received it, the tool used, the resource accessed, the authorization decision, and the final action. This creates a traceable delegation path and assists incident investigation.

The seven layers are designed to operate as an integrated security system rather than independent controls. Discovery identifies what must be protected, credential protection reduces exposure, authentication establishes identity, authorization limits access, runtime controls constrain behavior, monitoring detects misuse, and governance maintains accountability throughout the identity lifecycle.

Security Controls Against Major Machine Identity Attack Classes

The proposed framework must translate architectural principles into controls that address specific machine identity attack classes. Different attack types require different combinations of prevention, detection, restriction, and recovery. The objective is not to assume that compromise can always be prevented, but to reduce the probability of successful abuse and limit its consequences when it occurs.

API Key Leakage

API key leakage commonly occurs when credentials are stored in source-code repositories, configuration files, logs, or deployment scripts. Research has repeatedly shown that development environments can expose authentication secrets unintentionally (Basak et al., 2023; Meli et al., 2019).

The first control is to remove long-lived keys from application code and place secrets in dedicated credential-management systems. Automated repository scanning can identify exposed secrets before or after code is committed, but detection should be accompanied by immediate revocation and replacement because deleting the visible secret does not guarantee that it has disappeared from repository history.

API keys should also be scoped to the minimum required resources and, where possible, replaced by short-lived access tokens. Rate limits, source restrictions, request signing, and contextual policy checks can further reduce the usefulness of stolen credentials.

Service Account Compromise

Service accounts are particularly dangerous when they possess broad privileges or use persistent credentials. Controls should begin with assigning each service or application a distinct identity rather than sharing credentials across several systems.

Long-lived passwords should be replaced with managed identities, certificates, or temporary credentials where possible. Permissions should be reviewed regularly and

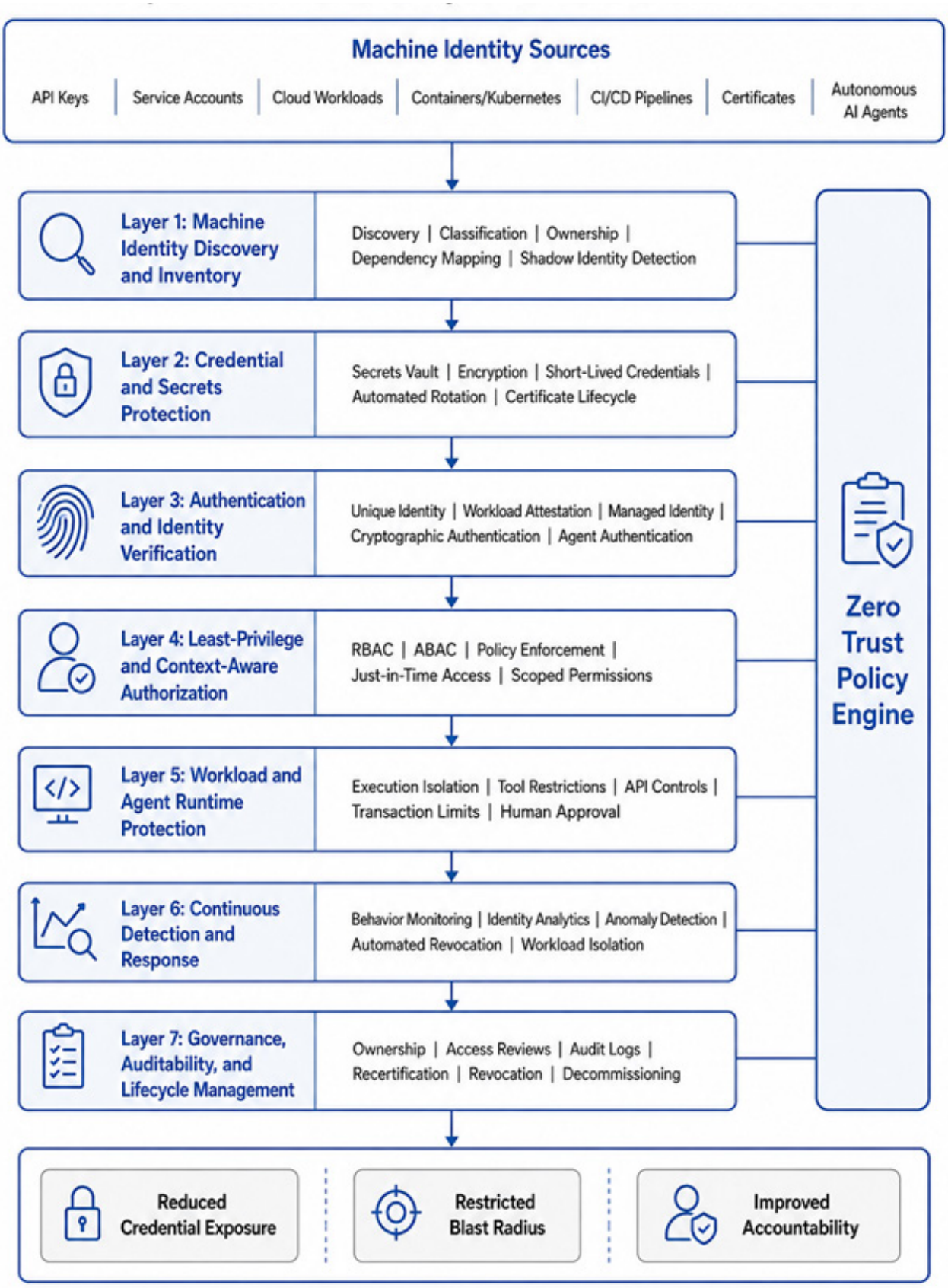

Fig1: Proposed Zero Trust Security Architecture for Machine Id

reduced to the smallest set necessary for the service function. Behavioral monitoring can also identify service-account compromise. Unexpected access to new resources, unusual execution environments, changes in normal activity, or attempts to use administrative functions can indicate misuse. When compromise is suspected, security systems should support rapid credential revocation and privilege suspension.

Workload Identity Theft

Workload identities can be stolen through exposed tokens, compromised containers, metadata services, or vulnerable runtime environments. Static credentials stored in deployment artifacts create particular risk because they can be copied and reused outside the intended workload.

Workload-bound authentication can reduce this exposure. Credentials should be linked to a specific runtime, service, or platform context and issued for limited periods. Workload attestation can provide additional assurance that requests originate from an approved environment.

In Kubernetes, restrictive RBAC policies are necessary because stolen service-account tokens can become more damaging when workloads possess excessive cluster permissions. Rostami (2023) emphasizes the importance of properly configured authorization within Kubernetes environments.

Credential Replay

Credential replay occurs when an attacker captures valid authentication material and reuses it to impersonate the legitimate identity. Bearer tokens are particularly susceptible because possession may be sufficient for access.

Short expiration periods reduce the replay window. Additional controls can bind credentials to a particular sender, workload, cryptographic key, or execution context. Contextual verification can also reject requests when the token is used from an unexpected environment.

Replay-resistant authentication is especially important for high-privilege machine identities because successful impersonation may otherwise appear indistinguishable from normal activity.

Machine Identity Privilege Escalation

Privilege escalation may result from excessive initial permissions, misconfigured roles, or the ability of one machine identity to assume another role. Preventive controls include least privilege, separation of duties, permission boundaries, and periodic access recertification.

Policy systems should also evaluate whether a machine is attempting actions outside its established operational function. A reporting service, for example, should not be able to modify production infrastructure simply because the underlying credential technically permits it.

Just-in-time privileges can further limit risk by reducing the amount of permanently available authority.

Lateral Movement

Lateral movement occurs when a compromised identity is used to access additional workloads or services. Network segmentation alone is not sufficient because machine communication increasingly crosses dynamic cloud and container environments.

Identity-aware microsegmentation and service-to-service authorization can limit the systems reachable by each workload. Chandramouli and Butcher (2023) support identity-centered access control for cloud-native environments where network location alone provides insufficient assurance.

Each downstream resource should independently authenticate and authorize the requesting machine. Compromise of one trusted service should therefore not provide automatic access to other systems.

AI Agent Abuse

Autonomous AI agents may be abused through manipulated prompts, malicious external content, excessive tool permissions, or inappropriate delegation. The primary defense is to separate an agent's ability to reason about an action from the authority to execute it.

Agents should receive only the tools required for their assigned role. Individual tools should enforce their own authorization rules, and sensitive operations should be checked against external security policies before execution.

Greshake et al. (2023) and Zhan et al. (2024) show that indirect prompt injection can manipulate tool-enabled systems. Consequently, untrusted content should not be allowed to determine authorization outcomes. Transaction limits, tool allowlists, data-access restrictions, and approval requirements provide additional boundaries around agent actions.

Compromised or Rogue Agent

A compromised or rogue agent presents greater risk when it possesses persistent access to several resources. Controls should therefore support rapid containment.

Agent actions should be continuously monitored for unusual tool selection, unexpected destinations, abnormal data access, policy violations, or attempts to expand privileges. A security control plane should be capable of suspending an agent identity, revoking its tokens, disabling tools, and terminating active sessions.

Human override should be available for high-impact systems. This control is particularly important when autonomous operations affect financial transactions, infrastructure configuration, sensitive information, or external communications.

The central principle across these attack classes is that no single control is sufficient. Secret protection reduces credential exposure but does not prevent misuse of legitimate authority. Least privilege limits damage but does not identify compromised identities. Monitoring can detect anomalies but cannot substitute for secure authentication.

Table1: Comparative Evaluation

Security Dimension	*Traditional Machine Identity Model*	*Zero Trust Machine Identity Framework*
Identity Discovery	Periodic or incomplete inventory	Continuous identity discovery and classification
Authentication	Static credentials commonly accepted	Unique, short-lived, context-bound authentication
Authorization	Broad standing privileges	Least privilege and contextual authorization
Credential Lifetime	Long-lived keys and passwords	Temporary and automatically rotated credentials
Workload Trust	Network or environment may influence trust	Every workload independently verified
Service Communication	Internal services often implicitly trusted	Explicit service-to-service authorization
Monitoring	Network and account activity monitored separately	Identity-centered continuous monitoring
Revocation	Often manual	Automated and policy-driven
AI Agent Access	General application permissions	Tool-level and action-level authorization
High-Risk Agent Actions	Limited formal controls	Risk-based human approval and execution limits
Lifecycle Governance	Fragmented across technologies	Unified ownership, review, rotation, and decommissioning

Effective machine identity security therefore depends on layered controls operating across prevention, authorization, detection, response, and governance.

Risk Assessment and Framework Evaluation

Machine Identity Risk Model

The evaluation of machine identities requires a risk model that captures differences between credential types, operational environments, privilege levels, and degrees of autonomy. A static API key and an autonomous agent cannot be assessed using credential exposure alone because their potential consequences differ substantially.

This study evaluates machine identity risk using seven dimensions

Machine Identity Risk = f(L, I, P, E, A, B, D)

- *where*
- L = Likelihood of compromise
- I = Potential impact
- P = Privilege level
- E = Credential exposure
- A = Degree of autonomy
- B = Blast radius
- D = Detection difficulty

Likelihood reflects the probability that an identity can be compromised or manipulated. Impact represents the expected consequences if abuse occurs. Privilege measures the authority available to the identity, while credential exposure examines how easily its authentication material can be obtained or reused.

Autonomy is included because identities capable of selecting and executing actions can create risks that conventional service accounts do not. Blast radius measures the number and sensitivity of resources accessible following compromise. Detection difficulty considers whether malicious behavior is likely to resemble legitimate machine activity.

Each dimension can be rated on a four-level scale

1 = Low, 2 = Moderate, 3 = High, 4 = Critical

The framework does not treat these ratings as empirical probabilities. Instead, they provide a consistent basis for comparing machine identities and prioritizing controls.

Representative Attack Scenarios

Framework evaluation is conducted through representative scenarios corresponding to the identity classes identified earlier.

Scenario 1: Leaked API Key

A static API key is accidentally committed to a public repository. Under a conventional model, the key may remain usable until manually revoked. Under the proposed framework, repository scanning, short credential lifetime, narrow scope, contextual access restrictions, and automated revocation reduce both exposure time and potential impact.

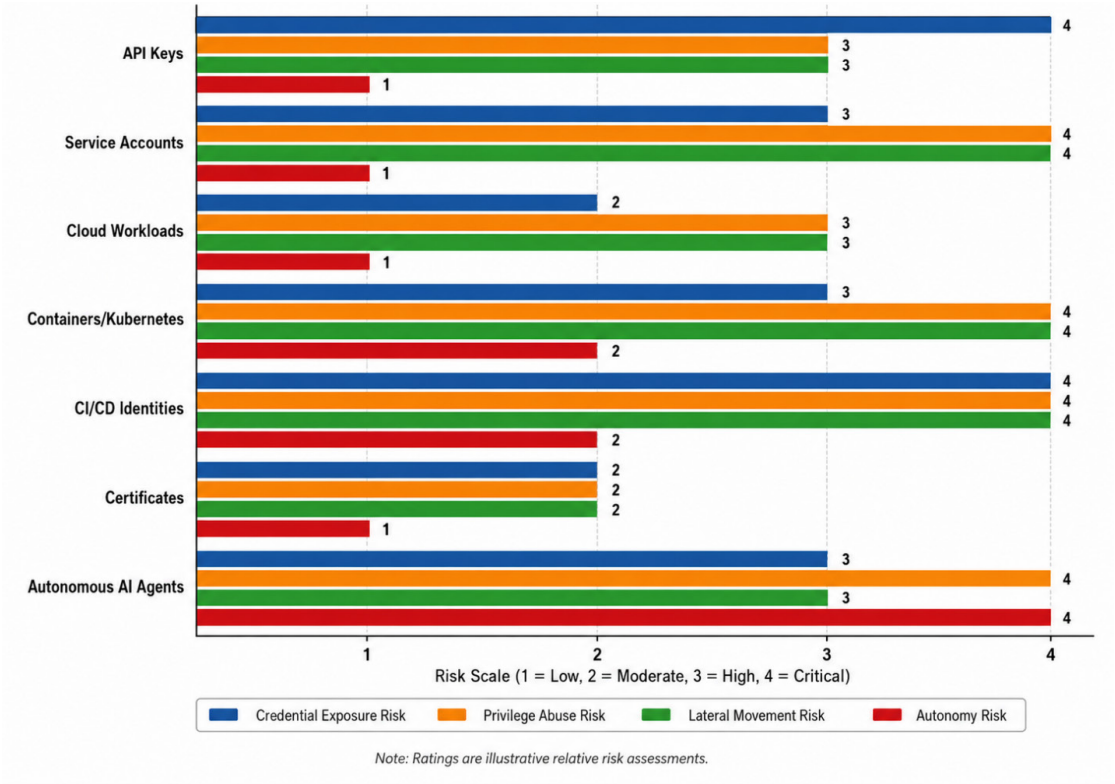

Fig1: Risk Expousure Across Major Machine Identity Types

Scenario 2: Compromised Service Account

An attacker obtains the credentials of an internal application service account. If the account has broad standing privileges, it may provide access to multiple resources. Under the proposed framework, unique identity assignment, least privilege, resource-specific authorization, behavioral monitoring, and rapid credential revocation restrict the attack path.

Scenario 3: Stolen Kubernetes Workload Identity

A workload token is extracted from a compromised container. The risk depends on whether the credential can be reused outside the original workload and what RBAC permissions are attached to it. Workload-bound authentication, short token lifetime, restrictive RBAC, and service-to-service policy enforcement reduce the usefulness of the stolen credential.

Scenario 4: CI/CD Credential Theft

A deployment token is exposed through a workflow or build environment. Such credentials can be highly sensitive because they may provide production access. The proposed framework limits this risk through ephemeral deployment credentials, separation of environments, protected execution contexts, least privilege, and detailed pipeline auditing. These controls address concerns identified in research on CI workflow security (Koishybayev et al., 2022).

Scenario 5: Autonomous Agent Credential Abuse

An attacker obtains a token assigned to an autonomous agent and attempts to use it directly. Short-lived credentials, agent-specific identity, contextual authentication, tool-level permissions, and continuous monitoring can restrict unauthorized use.

Scenario 6: Prompt-Injection-Driven Tool Misuse

A legitimate agent processes malicious external content that attempts to induce an unauthorized tool action. In this scenario, the agent's credential is not stolen. The failure occurs at the authorization and runtime levels. The proposed framework addresses the attack through separation of untrusted content from policy decisions, restricted tool permissions, action-level authorization, transaction thresholds, and human approval for high-impact operations. This scenario is consistent with the agent-security concerns identified by Greshake et al. (2023) and Zhan et al. (2024).

Security Evaluation Metrics

The framework can be evaluated using measurable security indicators that capture both identity hygiene and operational response. Relevant metrics include:

percentage of machine identities included in the enterprise inventory;

- percentage of identities with clearly assigned owners;
- percentage of workloads using short-lived credentials;

- mean and median credential lifetime;
- percentage of credentials rotated within policy;
- excessive privilege rate;
- percentage of orphaned identities;
- percentage of machine-to-machine connections subject to explicit authorization;
- unauthorized access detection rate;
- mean time to detect suspicious machine activity;
- mean time to revoke compromised credentials;
- percentage of autonomous agent actions subject to tool-level policy enforcement;
- percentage of high-risk agent actions requiring human approval; and
- average number of resources reachable after compromise of one identity.

These metrics allow organizations to evaluate whether security improvements are reducing exposure rather than simply increasing the number of deployed security controls.

Comparative Evaluation

The proposed framework can also be evaluated by comparing it with a traditional identity model.

Risk levels represent structured framework-based assessments derived from likelihood, impact, privilege, credential exposure, autonomy, blast radius, and detection difficulty. Values are not empirical incident frequencies

Framework Evaluation Outcome

The framework is expected to reduce machine identity exposure through three main mechanisms. First, it decreases the usefulness of stolen credentials by shortening credential lifetime, reducing privilege scope, and binding identities more closely to workloads and contexts. Second, it limits attack progression through explicit service authorization, microsegmentation, runtime restrictions, and resource-specific policy enforcement. Third, it improves detection and containment by connecting machine activity to identifiable workloads, services, pipelines, or agents.

The framework is particularly relevant to autonomous agents because traditional authentication does not determine whether an authenticated agent's intended action is appropriate. By separating agent identity from tool authorization and requiring additional controls for high-impact operations, the model addresses a security gap identified in recent agent research (Greshake et al., 2023; Ruan et al., 2024; Zhan et al., 2024).

The evaluation does not suggest that Zero Trust eliminates machine identity compromise. Rather, it changes the conditions under which compromise can produce broader damage. A stolen credential should have limited lifetime and scope, a compromised workload should have restricted paths to other systems, and a manipulated agent should encounter authorization boundaries before consequential actions are executed. These properties provide the basis for the governance and lifecycle controls discussed in the next section.

Governance and Machine Identity Lifecycle Management

Machine identity security depends on more than technical controls. Governance is required to ensure that every identity has a defined purpose, responsible owner, approved privileges, and a clear lifecycle. Without these controls, machine identities can remain active beyond their operational need, accumulate excessive permissions, or become disconnected from accountable teams. This risk is especially serious in cloud-native environments, where identities can be created automatically and at high volume.

Machine Identity Ownership

Every machine identity should be assigned to a responsible owner or team. Ownership establishes accountability for credential rotation, access reviews, policy compliance, and eventual decommissioning. The identity record should include its business purpose, associated workload, authorized resources, privilege level, and expected lifetime.

Identity Creation and Registration

Machine identities should be created through approved and traceable processes. Registration should occur before an identity is permitted to access enterprise resources. The process should capture identity type, owner, authentication method, resource scope, privilege level, and expiration requirements. Automated discovery can supplement registration by identifying identities created outside approved procedures.

Credential Issuance

Credential issuance should follow the principle of minimum necessary exposure. Short-lived and dynamically generated credentials should be preferred to persistent passwords or static API keys. Credentials should also be issued only to verified workloads or agents and protected through centralized secrets-management or identity services.

Privilege Assignment

Privileges should correspond directly to the operational function of the identity. Broad administrative access should be avoided unless it is strictly required. Least-privilege principles are particularly important for service accounts, CI/CD identities, and autonomous agents because these identities frequently operate without direct human supervision (Syed et al., 2022).

Continuous Monitoring

Machine identities should be monitored throughout their active lifecycle. Monitoring should capture authentication events, resource access, privilege use, unusual activity,

failed policy checks, and changes in expected behavior. Continuous verification supports the Zero Trust principle that previously authenticated identities should not be regarded as permanently trustworthy (Rose et al., 2020).

Credential Rotation

Credentials should be rotated according to risk, sensitivity, and credential type. High-privilege credentials should have shorter validity periods, while temporary workload and agent credentials should expire automatically when tasks or sessions end. Rotation should be automated wherever possible to reduce administrative error and long-term credential exposure.

Access Review and Recertification

Organizations should periodically review whether machine identities still require their assigned permissions. Access reviews should confirm ownership, purpose, privileges, resource dependencies, and recent usage. Identities with excessive or unused permissions should be modified or removed.

Revocation

Revocation should occur immediately when credentials are exposed, identities behave suspiciously, ownership becomes unclear, or operational need ends. Security systems should support automated revocation where compromise indicators exceed defined risk thresholds.

Machine Identity Decommissioning

Decommissioning should remove the identity, invalidate its credentials, terminate associated sessions, and eliminate unnecessary permissions and dependencies. Orphaned service accounts and forgotten credentials create persistent attack paths and should therefore be identified through continuous inventory and lifecycle reviews.

Autonomous AI Agent Governance

Autonomous agents require additional governance because they may act on behalf of users while interacting with multiple tools and systems. Each agent should have a registered identity, accountable owner, defined purpose, approved tool set, delegated authority, transaction limits, and clear escalation rules.

High-risk actions should require explicit approval, while audit records should preserve the relationship between the user, agent, tool, API, policy decision, and resulting action. Research on tool-integrated agents shows that legitimate agents may be manipulated into performing inappropriate actions, making authorization and auditability essential components of agent governance (Greshake et al., 2023; Zhan et al., 2024).

Discussion

The findings of this study show that machine identity security has become a distinct enterprise cybersecurity problem rather than a narrow extension of traditional IAM. API keys, service accounts, workloads, certificates, deployment identities, and autonomous agents operate differently from human users and often require continuous, automated authentication and authorization.

Machine Identity as a First-Class Cybersecurity Problem

Machine identities should be treated as independent security subjects. They often operate continuously, communicate at high frequency, and access sensitive systems without human interaction. A compromised identity can therefore provide attackers with persistent and difficult-to-detect access.

The literature on exposed secrets, container credentials, and CI/CD workflows confirms that machine credentials are frequently present across development and operational environments (Dahlmanns et al., 2023; Koishybayev et al., 2022; Meli et al., 2019).

From Secret Management to Identity-Centric Security

Protecting secrets is necessary but insufficient. A credential can remain confidential while the associated identity still creates risk through excessive privilege, weak ownership, poor monitoring, or inappropriate authorization.

Machine identity security should therefore focus on the full relationship between identity, credential, privilege, context, behavior, and resource access. This moves security beyond storing secrets securely toward controlling what each machine is permitted to do.

Zero Trust for Machines

Zero Trust provides a suitable foundation because it removes implicit trust and requires explicit authorization for protected resources. Applied to machine identities, this means that valid credentials alone should not provide unrestricted access. Workload identity, context, requested resource, privilege, and current risk should influence each access decision (Chandramouli & Butcher, 2023; Rose et al., 2020).

Security Implications of Autonomous AI

Autonomous agents extend the machine identity problem because they can interpret information and select actions rather than simply execute fixed application logic. Their security therefore depends on both credential protection and behavioral boundaries.

Research on prompt injection and agent manipulation shows that an authenticated agent may still perform an unauthorized action if it processes malicious instructions or possesses overly broad tool permissions (Greshake et al., 2023; Ruan et al., 2024; Zhan et al., 2024).

Relationship Between Autonomy, Privilege, and Risk

Autonomy and privilege should be evaluated together. An agent with limited access may present manageable risk even when highly autonomous, while an agent with broad administrative permissions can create a large blast radius.

Organizations should therefore apply stronger approval, monitoring, and authorization controls as the combination of autonomy and privilege increases.

Implications for Enterprise Security Architecture

Machine identity security requires closer integration between IAM, privileged access management, secrets management, API security, cloud security, workload protection, DevSecOps, and AI governance. Treating these areas separately can create gaps in ownership and policy enforcement.

The proposed framework addresses this fragmentation by placing machine identities under a common Zero Trust model that combines discovery, credential protection, authentication, authorization, runtime controls, monitoring, revocation, and governance.

Challenges and Limitations

Scale of Machine Identity Discovery

Large enterprises may operate vast numbers of machine identities across cloud platforms, applications, containers, certificates, and automation systems. Maintaining an accurate and current inventory can therefore be difficult.

Legacy Applications and Static Credentials

Older systems may depend on persistent passwords or embedded secrets and may not support short-lived credentials, modern workload identities, or dynamic authorization. Migration can therefore require significant technical changes.

Multi-Cloud Identity Fragmentation

Different cloud providers implement identity, token, role, and workload mechanisms differently. This makes it difficult to apply uniform policies across hybrid and multi-cloud environments.

Credential Rotation Complexity

Automated rotation can disrupt applications when dependencies are poorly documented or applications cannot retrieve updated credentials dynamically. Organizations must balance credential lifetime reduction with operational reliability.

Machine Identity Ownership Problems

Machine identities are often created by applications or development teams and may outlive the individuals who originally configured them. Assigning and maintaining ownership remains an organizational challenge.

Runtime Authorization Performance

Continuous authorization and context-aware policy evaluation may introduce additional processing requirements. High-volume machine-to-machine systems must therefore implement these controls without creating unacceptable latency.

Visibility Into Agent Behavior

Autonomous agents may perform multi-step tasks across several tools. Reconstructing why a specific action occurred can be difficult, particularly when intermediate reasoning or external inputs are not adequately logged.

Identity Federation Across Autonomous Agents

Multi-agent systems may require agents to delegate tasks and authority to one another. Establishing secure identity federation, delegation boundaries, and accountability across these interactions remains an open problem.

Standardization Challenges

Common standards for workload identities are developing, but autonomous agent identity and delegated authority remain less mature. Differences in implementation may hinder interoperability and consistent governance.

Limitations of the Proposed Framework

The framework developed in this study is conceptual and scenario-based. It has not been validated through large-scale enterprise deployment or controlled experimental testing. The proposed risk model is intended for structured comparison rather than prediction of real-world incident frequency. Future empirical studies are therefore required to measure its effectiveness across different organizational and technical environments.

Future Research Directions

Future research should examine how machine identity security can be strengthened as workloads and autonomous systems become more dynamic. Particular attention should be given to continuous authorization models that can reassess permissions during execution rather than relying on static policy decisions.

Research is also needed on cryptographic identities for autonomous agents, including methods for uniquely identifying agents, workloads, and temporary execution instances. Agent-to-agent authentication and secure delegation protocols should be investigated to prevent unauthorized transfer or amplification of privileges within multi-agent systems.

Another important area concerns automated least-privilege generation. Machine learning and policy-analysis

techniques could help organizations identify unused permissions, reduce excessive access, and recommend task-specific authorization boundaries. Behavioral methods could also support detection of compromised machine identities by identifying deviations from normal service or agent activity.

Future studies should further examine workload identity federation across multiple cloud providers, secure agent memory, identity-aware tool authorization, and methods for tracing delegation chains between humans, agents, tools, and enterprise resources.

Machine identity attack-path modeling also requires further development. Graph-based methods could help security teams determine how one compromised identity may enable lateral movement across interconnected systems. Automated revocation mechanisms based on verified risk indicators should also be evaluated.

Finally, there is a need for common standards that define autonomous agent identity, delegated authority, action provenance, approval requirements, and lifecycle governance. Such standards would improve interoperability and provide clearer security expectations as autonomous systems become more deeply integrated into enterprise operations.

Conclusion

Machine identities have become a major component of the modern enterprise attack surface. API keys, service accounts, cloud workloads, containers, certificates, CI/CD identities, and autonomous AI agents routinely access sensitive systems without direct human interaction. Their scale, persistence, privileges, and connectivity create security risks that cannot be addressed adequately through human-centered identity controls alone.

This study examined the principal threats associated with machine identity compromise, including credential leakage, impersonation, privilege escalation, lateral movement, persistent access, and misuse of autonomous agents. It also showed that protecting credentials alone is insufficient when legitimate identities can possess excessive authority or be manipulated into performing unauthorized actions.

The proposed Machine Identity Zero Trust Security Framework addresses these risks through continuous discovery, credential protection, strong authentication, least-privilege authorization, runtime controls, behavioral monitoring, automated revocation, and lifecycle governance. The framework extends Zero Trust principles beyond human users and applies them directly to workloads, services, automation systems, and autonomous agents.

As enterprise systems become increasingly automated, machine identity security should be treated as a core component of cybersecurity architecture. Effective protection will depend on reducing static trust, limiting persistent privileges, enforcing contextual authorization, and maintaining clear accountability throughout the complete machine identity lifecycle.

References:

[1] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero trust architecture*. NIST Special Publication 800-207. National Institute of Standards and Technology. doi:10.6028/NIST.SP.800-207.

[2] Chandramouli, R., & Butcher, Z. (2023). *A zero trust architecture model for access control in cloud-native applications in multi-location environments*. NIST Special Publication 800-207A. National Institute of Standards and Technology.

[3] Syed, N. F., Shah, S. W., Shaghaghi, A., Anwar, A., Baig, Z., & Doss, R. (2022). Zero Trust Architecture (ZTA): A comprehensive survey. *IEEE Access, 10*, 57143–57179. doi:10.1109/ACCESS.2022.3174679.

[4] He, Y., Huang, D., Chen, L., Ni, Y., & Ma, X. (2022). A survey on Zero Trust Architecture: Challenges and future trends. *Wireless Communications and Mobile Computing, 2022*, 6476274. doi:10.1155/2022/6476274.

[5] Teerakanok, S., Uehara, T., & Inomata, A. (2021). Migrating to Zero Trust Architecture: Reviews and challenges. *Security and Communication Networks, 2021*, 9947347. doi:10.1155/2021/9947347.

[6] Ferretti, L., Magnanini, F., Andreolini, M., & Colajanni, M. (2021). Survivable zero trust for cloud computing environments. *Computers & Security, 110*, 102419. doi:10.1016/j.cose.2021.102419.

[7] Adahman, Z., Malik, A. W., & Anwar, Z. (2022). An analysis of zero-trust architecture and its cost-effectiveness for organizational security. *Computers & Security, 122*, 102911. doi:10.1016/j.cose.2022.102911.

[8] Phiayura, P., & Teerakanok, S. (2023). A comprehensive framework for migrating to Zero Trust Architecture. *IEEE Access, 11*, 19487–19511. doi:10.1109/ACCESS.2023.3248622.

[9] Kang, H., Liu, G., Wang, Q., Meng, L., & Liu, J. (2023). Theory and application of Zero Trust security: A brief survey. *Entropy, 25*(12), 1595. doi:10.3390/e25121595.

[10] Potla, R. B. (2024). A SOX/ITAR-aligned global ERP template for multi-plant manufacturers: Governance patterns and controls. J Artif Intell Mach Learn & Data Sci, 2(2), 3222-3232.

[11] Fernandez, E. B., & Brazhuk, A. (2024). A critical analysis of Zero Trust Architecture (ZTA). *Computer Standards & Interfaces, 89*, 103832. doi:10.1016/j.csi.2024.103832.

[12] Azad, M. A., Abdullah, S., Arshad, J., Lallie, H., & Ahmed, Y. H. (2024). Verify and trust: A multidimensional survey of zero-trust security in the age of IoT. *Internet of Things, 27*, 101227. doi:10.1016/j.iot.2024.101227.

[13] Ramezanpour, K., & Jagannath, J. (2022). Intelligent zero trust architecture for 5G/6G networks: Principles, challenges, and the role of machine learning in the context of O-RAN. *Computer Networks, 217*, 109358. doi:10.1016/j.comnet.2022.109358.

[14] Buck, C., Olenberger, C., Schweizer, A., Völter, F., & Eymann, T. (2021). Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust. *Computers & Security, 110*, 102436.

[15] Chen, B., Qiao, S., Zhao, J., Liu, D., Shi, X., Lyu, M., Chen, H., Lu, H., & Zhai, Y. (2021). A security awareness and protection system for 5G smart healthcare based on Zero-Trust Architecture. *IEEE Internet of Things Journal, 8*(13), 10248–10263. doi:10.1109/JIOT.2020.3041042.

[16] KUNAPARAJU, C. (2024). Ethical and Legal Challenges of AI-Based Surveillance in National Security Operations. Journal of Computational Analysis and Applications (JoCAAA), 33(08), 8602-8625.

[17] Teng, W. L., & Rasmussen, K. (2023). Actions speak louder

than passwords: Dynamic identity for machine-to-machine communication. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES 2023)*, 1–11. doi:10.1145/3600160.3600165.

[18] Deochake, S. (2022). Identity and access management framework for multi-tenant resources in hybrid cloud computing. In *Proceedings of the 2022 ACM Southeast Conference*. doi:10.1145/3538969.3544896.

[19] Rostami, G. (2023). Role-based Access Control (RBAC) authorization in Kubernetes. *Journal of ICT Standardization, 11*(3), 237–260. doi:10.13052/jicts2245-800X.1132.

[20] Shamim, M. S. I., Bhuiyan, F. A., & Rahman, A. (2020). XI commandments of Kubernetes security: A systematization of knowledge related to Kubernetes security practices. In *2020 IEEE Secure Development Conference (SecDev)*, 58–64.

[21] Meli, M., McNiece, M. R., & Reaves, B. (2019). How bad can it Git? Characterizing secret leakage in public GitHub repositories. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2019)*.

[22] Potla, R. B. (2024). Optimizing extended warehouse management for make-to-order plants: Slotting, wave picking, and yard orchestration at scale. Journal of Computer Science and Technology Studies, 6(3), 181-192.

[23] Sinha, V. S., Saha, D., Dhoolia, P., Padhye, R., & Mani, S. (2015). Detecting and mitigating secret-key leaks in source code repositories. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 396–400. doi:10.1109/MSR.2015.48.

[24] Basak, S. K., Neil, L., Reaves, B., & Williams, L. (2023). What challenges do developers face about checked-in secrets in software artifacts? In *Proceedings of the 45th International Conference on Software Engineering*, 1635–1647. doi:10.1109/ICSE48619.2023.00141.

[25] Dahlmanns, M., Sander, C., Decker, R., & Wehrle, K. (2023). Secrets revealed in container images: An Internet-wide study on occurrence and impact. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. doi:10.1145/3579856.3590329.

[26] Koishybayev, I., Nahapetyan, A., Zachariah, R., Muralee, S., Reaves, B., Kapravelos, A., & Machiry, A. (2022). Characterizing the security of GitHub CI workflows. In *31st USENIX Security Symposium*, 2747–2764.

[27] Rahman, M. R., Imtiaz, N., Storey, M. A., & Williams, L. (2022). Why secret detection tools are not enough: It's not just about false positives, an industrial case study. *Empirical Software Engineering, 27*(3), 59. doi:10.1007/s10664-021-10109-y.

[28] Krause, A., Klemmer, J. H., Huaman, N., Wermke, D., Acar, Y., & Fahl, S. (2022). *Committed by accident: Studying prevention and remediation strategies against secret leakage in source code repositories*. arXiv:2211.06213.

[29] Potla, R. (2023). Designing a BTP-centric integration mesh for shop-floor IoT, MES and ERP in discrete manufacturing. Journal of Artificial Intelligence, Machine Learning and Data Science, 1(2), 1-8.

[30] Fett, D., Küsters, R., & Schmitz, G. (2016). A comprehensive formal security analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 1204–1215. doi:10.1145/2976749.2978385.

[31] Philippaerts, P., Preuveneers, D., & Joosen, W. (2022). OAuch: Exploring security compliance in the OAuth 2.0 ecosystem. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 460–481. doi:10.1145/3545948.3545955.

[32] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 2023 ACM Workshop on Artificial Intelligence and Security*, 79–90. doi:10.1145/3605764.3623985.

[33] Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C. J., & Hashimoto, T. (2024). Identifying the risks of LM agents with an LM-emulated sandbox. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*.

[34] Zhan, Q., Liang, Z., Ying, Z., & Kang, D. (2024). InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 10471–10506. doi:10.18653/v1/2024.findings-acl.624.