What Is Non-Human Identity (NHI)? The Identity Class That Outnumbers Humans 10-to-1 or More

Definitions / Last updated 2026-06-11 / https://www.scrambleid.com/learn/what-is-non-human-identity

In one sentence: Non-human identity (NHI) is the umbrella term for any identity that is not a human user, service accounts, workloads, microservices, CI/CD pipelines, AI agents, MCP servers, devices, IoT, and bots, and it is now the dominant identity attack surface in modern enterprises because NHIs outnumber humans by an order of magnitude and are typically authenticated with long-lived shared secrets that leak, get reused, and persist beyond their useful life.

TL;DR (canonical)

- **Definition:** Identities that are not human end-users.
- **Scope:** Service accounts, workloads, microservices, CI/CD pipelines, AI agents, MCP servers, connected devices, IoT, bots.
- **Volume:** NHIs typically outnumber human identities by 10:1 or more, up to 50:1 in some estimates, in modern enterprises.
- **Dominant failure mode:** Long-lived shared secrets (API keys, passwords, SSH keys) that leak, get reused, or persist past their useful life.
- **The replacement architecture:** Cloud-native workload identity (AWS IRSA, GCP Workload Identity Federation, Azure Managed Identity, SPIFFE/SPIRE) plus sender-constrained tokens (RFC 8705 mTLS, RFC 9449 DPoP) plus short-lived credentials brokered at runtime (RFC 7523 JWT assertions).
- **The new frontier:** AI agents and MCP servers as first-class identities, distinct from the user delegating to them.

What counts as an NHI

Category	Examples
Service accounts	The "service account" your application uses to talk to a database, message queue, or third-party API
Workloads	Containers, Kubernetes pods, serverless functions, VMs running specific applications

Category	Examples
Microservices	Internal services calling each other in a service mesh
CI/CD pipelines	GitHub Actions runners, GitLab CI, Jenkins, CircleCI, Buildkite, etc.
AI agents	Autonomous or semi-autonomous LLM-driven systems acting on behalf of users
MCP servers	Model Context Protocol servers acting as tool brokers between AI agents and APIs
Bots	Slack bots, GitHub bots, automation accounts
Connected devices	IoT, medical devices, industrial sensors, edge devices
Robots / RPA	Robotic Process Automation tools authenticating into enterprise systems
Third-party integrations	OAuth-connected SaaS-to-SaaS integrations

The defining property is "not a human end-user." If a credential is meant to authenticate a process, a service, a device, or any other non-person, it is an NHI.

Why NHI is now the dominant identity attack surface

Volume. A modern enterprise easily has tens of thousands of NHIs. Cloud workloads alone account for thousands. Add CI pipelines, integrations, devices, and AI agents, and the count outpaces humans by orders of magnitude.

Sprawl. Service-account credentials accumulate in source code, environment variables, vault entries, terraform state, CI configurations, and developer laptops. Most enterprises do not have a complete inventory.

Long-lived secrets are the default. Service accounts have historically used long-lived API keys or passwords. The credential is created once and used for years. When it leaks, it remains valid.

Credentials are not bound to context. A leaked API key can be replayed from anywhere, by any actor. There is no equivalent of "this credential only works from this workload, on this network, in this cloud account."

The attacker has noticed. High-profile breaches (CircleCI, Cloudflare-related, Okta support tooling, others) have started with, or been amplified by, leaked or compromised NHI credentials. The economic return on attacking NHIs is now greater than on attacking phished human credentials in many enterprise contexts.

Compliance has not kept up. Most regulatory frameworks were written for human authentication. NHI controls are increasingly explicit (FedRAMP IA controls cover non-human identities; PCI DSS v4.0.1 explicitly requires authentication for "all access" including service-to-service), but the operational reality is far behind.

The architecture that replaces long-lived secrets

Cloud-native workload identity

The most leveraged change. Cloud providers offer mechanisms where a workload can prove its identity to the cloud's IAM without holding any persistent secret:

Cloud	Mechanism
AWS	IAM Roles for Service Accounts (IRSA) for EKS workloads; IAM Roles Anywhere for off-cloud workloads; STS AssumeRole with web identity for federated workloads
GCP	Workload Identity Federation; Workload Identity for GKE; Service Account impersonation via short-lived tokens
Azure	Managed Identity (system-assigned and user-assigned); Microsoft Entra ID Workload Identity for AKS
Cross-cloud / hybrid	SPIFFE/SPIRE for workload attestation across heterogeneous environments

The pattern: the workload's runtime (the cluster, the cloud-managed service) attests its identity to the cloud's IAM. The cloud issues a short-lived credential scoped to what the workload needs. There is no persistent secret in source control or in a vault.

Sender-constrained tokens

Even with workload identity, service-to-service calls across boundaries need credentials in flight. The bar:

- **RFC 8705 mTLS**: access tokens bound to the client's TLS certificate. The token is only valid when presented over a mutually-authenticated TLS connection with that certificate.
- **RFC 9449 DPoP**: access tokens bound to a client's public key, with each request signed by the corresponding private key.
- **RFC 7523 JWT client assertions**: the client authenticates to the token endpoint with a signed JWT instead of a shared secret.

These standards make a leaked token useless without the corresponding private key, key pair, or certificate.

Short-lived credentials and OIDC federation

CI/CD pipelines should not hold persistent admin keys. The pattern:

- GitHub Actions: native OIDC token federated to AWS IAM, GCP Workload Identity, Azure federated credentials.
- GitLab CI/CD: ID tokens with the same federation pattern.
- Jenkins, CircleCI, Buildkite: OIDC support varying by maturity.

The pipeline authenticates to the cloud at the moment of action with a credential whose lifetime is the duration of the job.

For deeper coverage, see M2M Authentication Without Secrets and Machine Identity (PoP, DPoP, mTLS).

NHI for AI agents and MCP servers

AI agents introduce new NHI patterns. The two failure modes are:

1. The agent reuses the user's bearer token at full scope. The agent gets the user's session, calls APIs, and operates with the user's full privileges, even if the user only intended to delegate a narrow task. Prompt injection or tool misuse then becomes a full account-takeover-equivalent compromise.

2. The MCP server is treated as a passive intermediary with a shared API key. Anyone who reaches the MCP server's endpoint with the shared key can use it as a tool broker.

The architecture that addresses both:

- **The agent has its own identity.** Attested at instantiation; verifiable at every action.
- **The user's delegation is explicit and scoped.** Short-lived, narrow scope, signed.
- **Every action presents three identities:** the agent, the user delegation, the target resource.
- **The MCP server is a first-class identity.** Sender-constrained credential of its own; not a shared key.
- **Authorization decisions are made on the full context**, not on the user's bearer token alone.

For the full pattern, see AI Agent Authentication and the AI Agent Tool Access Playbook.

Common NHI failure modes

1. **Hardcoded credentials in source code.** Trivially discoverable in repository history, public GitHub leaks, and CI logs.
2. **Long-lived API keys in environment variables.** Persist beyond the workload's lifetime, exposed via diagnostic dumps and credential exfiltration.
3. **Vault entries with no expiration.** A vault is a credential store, not a security guarantee. Long-lived secrets in vaults are still long-lived.
4. **Service accounts with broad privileges.** "Make it work" provisioning produces accounts with cluster-admin or full-API scope when narrow scope would suffice.
5. **Service accounts that outlive the system they were created for.** A decommissioned application's service account that still works months later.
6. **CI/CD pipeline credentials with admin scope.** Once leaked, the entire production environment is reachable.

7. **Webhook secrets that never rotate.** A leaked webhook HMAC key remains valid until manually rotated.
8. **Inherited credentials via clone.** A new workload deployed by copying configuration inherits the credentials of the workload it was copied from.
9. **AI agents using the user's full bearer token.** The agent's authorization is the user's authorization, with no scope reduction.
10. **MCP servers using shared API keys.** Any client that finds the endpoint can use it.

How to start fixing NHI in an enterprise

A practical sequence:

1. **Inventory.** Build a real inventory of NHIs. Most enterprises start with vault audits, IAM policy mining, and CI/CD configuration analysis. The inventory is rarely complete on the first pass; iterate.
2. **Classify by blast radius.** Production-impacting service accounts and admin-equivalent CI credentials first.
3. **Identify long-lived secrets.** Anything that doesn't expire automatically is in scope for replacement.
4. **Migrate to cloud-native workload identity** for in-cloud paths. This is the highest-leverage move.
5. **Migrate cross-boundary paths to sender-constrained tokens.** mTLS or DPoP, depending on the calling pattern.
6. **Migrate CI/CD to OIDC federation.** GitHub, GitLab, and most modern CIs support it.
7. **Address AI agents and MCP servers.** First-class identity, scoped delegation, sender-constrained credentials.
8. **Build the audit narrative.** Every NHI authentication event signed and queryable.
9. **Rotate and decommission.** Long-lived credentials don't disappear when you stop using them; revoke them.
10. **Continuous attestation.** NHIs attest their identity continuously, not just at registration.

Key Takeaway

Non-human identity (NHI) is the umbrella term for identities that are not human users, including service accounts, workloads, microservices, CI/CD pipelines, AI agents, MCP servers, connected devices, and bots. NHIs typically outnumber humans by 10:1 or more in modern enterprises and are now the dominant identity attack surface because long-lived shared secrets are the default. The architecture that replaces long-lived secrets: cloud-native workload identity (AWS IRSA, GCP Workload Identity Federation, Azure Managed Identity, SPIFFE/SPIRE), sender-constrained access tokens (RFC 8705 mTLS, RFC 9449 DPoP), short-lived credentials brokered at runtime, and OIDC

federation between CI/CD and clouds. AI agents and MCP servers are the new frontier and require first-class identity rather than reusing the human user's bearer token.

FAQ

What is non-human identity (NHI)?

Non-human identity refers to identities that are not human users: service accounts, workloads, microservices, CI/CD pipelines, AI agents, MCP servers, devices, IoT, bots, and any other non-person entity that authenticates to a system. NHIs typically outnumber human identities by 10:1 or more in modern enterprises, and they are the dominant credential-management failure mode because long-lived shared secrets are the default for most NHI authentication.

How is NHI different from machine identity?

The terms overlap and are often used interchangeably. "Machine identity" historically referred to TLS certificates for servers and devices. "Non-human identity" is a broader umbrella covering all identities that are not human users, including the older machine-identity scope plus service accounts, workloads, AI agents, MCP servers, and CI/CD pipelines. NHI is the more current term in CISO and analyst conversations.

Why are long-lived service-account secrets a problem?

Long-lived shared secrets accumulate in source code, vault entries, environment variables, CI/CD configurations, and developer laptops. They get leaked, exfiltrated, or harvested. Once leaked, they remain valid for the duration of their lifetime, which is often months or years. They are not bound to a specific workload or network, so a leaked credential can be replayed from anywhere. Multiple high-profile breaches (CircleCI, Okta support tooling, others) have started with, or been amplified by, leaked long-lived NHI credentials.

What replaces long-lived secrets for NHI?

Three patterns: cloud-native workload identity (AWS IRSA, GCP Workload Identity Federation, Azure Managed Identity, SPIFFE/SPIRE for cross-cloud), sender-constrained access tokens (mTLS per RFC 8705 or DPoP per RFC 9449), and short-lived credentials brokered at runtime (JWT client assertions per RFC 7523, OIDC federation between CI providers and clouds). The pattern is: no persistent secret in source control or vault; credentials are issued at the moment of action with a lifetime measured in minutes.

Are AI agents non-human identities?

Yes. An AI agent acting on behalf of a user is a non-human identity that needs its own authentication. Treating the agent as "whatever credential the user happened to have" is the most common failure

mode and the next breach class. Mature agent architectures bind agent identity, user delegation, and resource scope at every action, with the agent presenting all three to the authorization service. Same applies to MCP (Model Context Protocol) servers.

How do NHIs fit into Zero Trust?

Zero Trust extends to non-human identities. Every authentication, including service-to-service, must be explicit and verified. The CISA Zero Trust Maturity Model includes identity as Pillar 1 and applies to NHIs as much as to humans. Practical implementation: every NHI has a verifiable identity, every service-to-service call uses a sender-constrained credential, every credential is short-lived, and every authentication event is auditable.

References (public)

- RFC 9700 (OAuth 2.0 Security Best Current Practice): https://www.rfc-editor.org/rfc/rfc9700.html
- RFC 8705 (OAuth 2.0 Mutual-TLS): https://datatracker.ietf.org/doc/html/rfc8705
- RFC 9449 (OAuth 2.0 DPoP): https://www.rfc-editor.org/rfc/rfc9449.html
- RFC 7523 (JWT Client Authentication): https://datatracker.ietf.org/doc/html/rfc7523
- SPIFFE/SPIRE: https://spiffe.io/
- AWS IAM Roles for Service Accounts (IRSA): https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts.html
- Google Cloud Workload Identity Federation: https://cloud.google.com/iam/docs/workload-identity-federation
- Azure Managed Identities: https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/
- CISA Zero Trust Maturity Model: https://www.cisa.gov/zero-trust-maturity-model

---

Related reading

- M2M Authentication Without Secrets
- Machine Identity (PoP, DPoP, mTLS)
- AI Agent Authentication
- AI Agent Tool Access Playbook
- What Is Phishing-Resistant MFA?
- What Are NIST AAL Levels?