CREMIT RESEARCH

# The NHI Security Playbook

A practitioner's reference for finding, fixing, and governing non-human identities.

2026 Q2 Edition · cremit.io

# Contents



> **WHO THIS IS FOR**
>
> Security engineers, DevSecOps leads, CISOs, and platform teams who've realized that API keys, service accounts, and OAuth tokens outnumber their humans 45:1 — and no one has been putting this on the governance roadmap.

PART 1

# What is the NHI security problem?

A non-human identity is any credential, key, or identifier that authenticates a workload rather than a person. The category is broader than most people realize.

- **Service accounts** — IAM users and roles that exist to run jobs
- **API keys** — static strings issued by Stripe, Slack, OpenAI, etc.
- **Access tokens** — OAuth bearer tokens, JWTs, refresh tokens
- **Machine certificates** — mTLS, code-signing, device certs
- **SSH keys** — deploy keys, CI runner keys
- **Cloud workload identities** — AWS IAM roles, GCP service accounts, K8s SAs
- **Webhook secrets**, **DB credentials**, **AI agent credentials**

The common thread: a human is not typing the password. That one fact breaks a lot of assumptions — you cannot require MFA on an API call, and you usually cannot tell from a log who is actually using the key.

## Why NHIs matter right now

Three forces compounded over five years: **SaaS sprawl** (the average enterprise runs 130+ apps, each with its own NHI footprint), **microservices** (hundreds of service-to-service credentials where a monolith had one), and **AI agents** (a new NHI class that barely existed two years ago).

> **WHAT 45:1 LOOKS LIKE**
>
> A 2,000-person engineering org running on AWS, GitHub, and ~80 SaaS tools typically has **~2,000 humans, ~40,000 SaaS grants, and 90,000+ active NHIs**. Conservative estimate.

## The numbers you need to know

**23.8M**
secrets detected on public GitHub in 2024 (+25% YoY) — GitGuardian State of Secrets Sprawl 2025

**5 yrs**
Toyota's exposed access key sat in a public repo before discovery, exposing 296,019 customer records

**70%+**
of cloud breaches in the last 3 years involved a compromised NHI somewhere in the chain

**CASE: TOYOTA TIMELINE**

An access key committed to a public repository. Secret scanning didn't catch it. Authenticated to a customer-facing data service. Remained valid for roughly five years. When finally disclosed, remediation required notifying ~300,000 people. Every stage of the Kill Chain in Part 2 appears in this single incident.

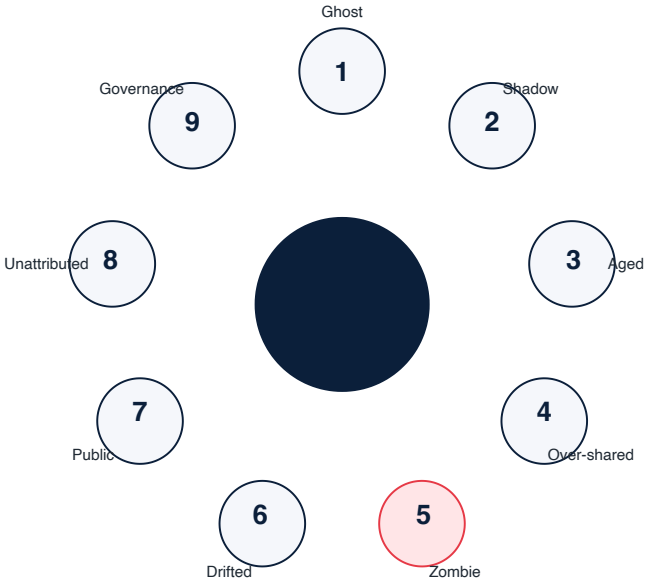
Continued in full playbook (pages 4-5): Uber 2016, median time-to-rotate, and the Stripe/Slack case analysis.

PART 2

# The NHI Kill Chain

The NHI Kill Chain is a taxonomy of the nine most common ways non-human identities fail. It's a pattern language derived from real incidents, not a theoretical threat model.

Each pattern has a name, a one-sentence definition, an explanation of why it happens, detection signals for your environment, and a key question to bring to your next review.




The nine NHI Kill Chain patterns. Pattern 5 (Zombie) highlighted — the most under-detected failure mode.

## Patterns 1 through 3

### 1 Ghost Key

*A credential exists in your environment that nobody created, owns, or remembers.*

**Why it happens:** M&A, contractor work, one-off PoCs, and deprecated projects leave credentials that outlive the context that created them.

**Detection:** IAM users with no owner field; service accounts with no ticket or runbook; keys not rotated since before your current SRE team joined.

> *"For every privileged credential in our environment, can a single named human tell me today what it does and why it exists?"*

### 2 Shadow Key

*A credential exists that your security and platform teams do not know about.*

**Why it happens:** Created outside approved workflows — personal access tokens, self-minted webhook secrets, business-side-authorized OAuth grants.

**Detection:** PATs with broad org-wide scope, unreviewed OAuth grants in SaaS admin consoles.

> *"What is our single source of truth for active NHIs, and how confident are we it's complete?"*

### 3 Aged Key

*A credential has been valid for so long that its original security posture is no longer credible.*

**Why it happens:** Rotation is painful. Teams defer it. Once a key has been live for a year it tends to stay live indefinitely.

> *"What is our maximum acceptable age for a production credential, and how many exceed it today?"*

## Patterns 4 through 6

### 4 Over-shared Key

*A single credential duplicated across so many systems that revocation is operationally impossible.*

> **FIELD NOTE**
> A Series B fintech's production Stripe secret was present in 14 locations — including two ex-employee laptops, a Slack pinned message from 2023, a Confluence runbook, and a contractor's Dockerfile. Rotation coordination took six weeks.

> *"If we rotated our most important production secret today, how many systems would break?"*

### 5 Zombie Key (the most under-detected)

*A credential is reported as resolved in a scanner, but is still valid at the provider.*

**Why it happens:** Engineer removes the string from source, closes the alert, never revokes at the IdP. Scanner is happy. The key is still accepted.

> *"When we close a leaked-credential finding, is revocation at the provider a required step?"*

### 6 Drifted Key

*Provisioned with tight scope, but effective permissions have silently expanded.*

**Detection:** IAM roles with more policies than at creation; service accounts with capabilities unrelated to their stated purpose.

> *"When did we last review the effective permissions of our top 20 most-used service accounts?"*

## Patterns 7 through 9

### 7 Public Key

*Credential in a public-facing location a scanner can find.*

Accidental git pushes, misconfigured buckets, public Postman collections, client-side JS bundles embedding server-side keys, compiled mobile apps. Most organizations think they've solved this. 23.8M secrets on public GitHub in 2024 alone says otherwise.

### 8 Unattributed Key

*Active credential, but observability cannot tell you which workload is using it or who is responsible.*

> *"If our highest-privilege NHI were used maliciously in the next hour, how long would identifying the workload and owner take?"*

### 9 Governance Gap

*Systemic absence of NHI ownership.*

Most organizations have no single accountable owner for NHI security. Human identity sits with IAM, endpoint with security, cloud with platform — and NHIs fall in the gaps. Without an owner, the preceding eight patterns can't be systematically fixed.

# The NHI Kill Chain at a glance

One-page reference. Print and pin it.

| # | Pattern | Definition | Primary detection signal |
|---|---|---|---|
| 1 | Ghost Key | Credential with no known owner | Missing owner field, decommissioned project names |
| 2 | Shadow Key | Credential security/platform doesn't know about | Unreviewed OAuth grants, org-scope PATs |
| 3 | Aged Key | Credential older than the trust window | Creation date > 12 months, no rotation history |
| 4 | Over-shared Key | Credential duplicated across too many systems | Same secret hash across multiple stores |
| 5 | Zombie Key | Scanner says resolved, provider says valid | "Resolved" alerts without IdP revocation |
| 6 | Drifted Key | Scope expanded silently past original intent | IAM policies added after initial provisioning |
| 7 | Public Key | Credential in a scanner-reachable public location | Any public-source secret scanning alert |
| 8 | Unattributed Key | Active credential with no workload attribution | Auth logs missing workload metadata |
| 9 | Governance Gap | No named owner for NHI security | Responsibility sits in a gap between teams |

PART 3

# The "Out of Scope" problem

Bug bounty programs were designed for exploit-chain vulnerabilities. Credential exposure doesn't fit CVSS. When researchers report an exposed organizational credential, a predictable thing happens: it's classified "out of scope" and dismissed.

> **CASE 1 — SLACK BOT TOKEN**
> Sat in a public GitHub repo for **three years**. Broad channel and file access across the organization. Reported via official bug bounty. Classification: **out of scope**. Action: the organization quietly revoked the key and conducted a broader review anyway.

> **CASE 2 — ASANA ADMIN API KEY**
> Exposed for **two years**, full workspace read/write. Reported via official bug bounty. Classification: **out of scope** (asset originated from a consolidated entity). Action: also revoked.

## Why this keeps happening

1. **CVSS doesn't fit.** No exploit chain, no attack complexity, no privileges-required — the credential is the access.
2. **"It's just a misconfiguration."** But RCE is also the result of bad code, and we evaluate RCE by impact, not cause. The double standard doesn't hold.
3. **Time is invisible.** Three years of exposure scores the same as a credential leaked yesterday.

## The NHI Exposure Severity Index

A complementary framework to CVSS, built for credentials discovered in the wild. Six axes, each scored 1 to 5.

| Axis | What it measures | Slack case |
|---|---|---|
| Privilege Scope | Read-only → unrestricted admin | 4 |
| Cumulative Risk Duration | <24h → multi-year | 5 |
| Blast Radius | Single resource → multi-org | 4 |
| Exposure Accessibility | Auth-required → publicly indexed | 5 |
| Data Sensitivity | Public info → regulated PII/secrets | 4 |
| Lateral Movement Potential | Isolated → cross-service harvestable | 5 |

**Slack case total: 27/30.** A finding with three or more axes at 4 or 5 is Severity 1 regardless of how bug bounty triage classifies it.

## Advocating for better scope

1. Explicitly include organizational credential exposure in your program scope.
2. Route credential-exposure findings through a separate pipeline using the NHI Index, not CVSS.
3. Publish your rotation SLA — this is the most effective public signal that credential exposure is taken seriously.

PART 4

# Korean ISMS-P crosswalk

ISMS-P의 강화 인증기준은 2027 의무화가 다가오면서, 크리덴셜 관리·접근 통제·감사 로깅에서 NHI Kill Chain 대응 체크리스트와 놀라울 만큼 겹칩니다.

| Kill Chain Pattern | 가장 가까운 ISMS-P 컨트롤 영역 |
|---|---|
| Ghost Key | 2.5.1 사용자 계정 관리 — 자산 소유자의 유일 식별 |
| Shadow Key | 2.5.2 사용자 식별 및 인증 — 인벤토리 완전성 |
| Aged Key | 2.5.4 비밀번호 관리 — 크리덴셜 생애주기·회전 |
| Over-shared Key | 2.5.3 사용자 접근권한 관리 — 최소권한·분리 |
| Zombie Key | 2.9.4 로그 및 접속기록 관리 + 2.5.1 — 폐기 증거 |
| Drifted Key | 2.5.5 접근통제 관리 — 주기적 권한 리뷰 |
| Public Key | 2.11.5 외부자 보안 + 2.10.6 업무용 단말기기 보안 |
| Unattributed Key | 2.9.2 성능·장애관리 + 2.9.4 — 로그 귀속 |
| Governance Gap | 1.1.1 경영진의 참여 + 1.1.2 최고책임자의 지정 |

## 2026 preparation calendar

- **Q2 2026 (now):** Stand up NHI inventory. Imperfect is fine — auditors care about trend.
- **Q3 2026:** Rotation for top 20 most-privileged NHIs, runbook documented.
- **Q4 2026:** Close attribution gap in audit logs.
- **Q1 2027:** Internal audit against the mapping above, remediate before external audit.

PART 5

# The 30-60-90 action playbook

## Week 1 — Inventory

Export IAM from every cloud, active API keys from top 10 SaaS, OAuth grants from your IdP, deploy keys and PATs from source control. One spreadsheet. Single list you can hand your CISO on Friday. *Completeness > polish.*

## Week 2 — Identify the unowned

For each entry, name a single human owner. Flag anything you can't within 48 hours. Produce the unowned credential list — your immediate risk surface.

## Week 3 — Age and rotation

Sort by creation date. Every credential > 12 months: confirm rotation once, or schedule it. Start a rotation queue of three per week.

## Week 4 — CI/CD and exposure detection

Secret scanning on push *and* history. Require revocation-at-provider as alert closure condition (kills the zombie pattern). CI policy blocking literal credentials in PRs.

## 30-60-90 roadmap

- **30 days:** inventory, unowned list, zombie-pattern closure hardening, aged-key rotation queue.
- **60 days:** 100% production NHI ownership, OAuth review process, top-20 rotation runbooks, external monitoring.
- **90 days:** drift detection, attribution gap closed, bug bounty scope updated, weekly Kill Chain dashboard.

> **SUCCESS METRICS THAT MATTER**
> Unowned credentials trending to 0 · Privileged credentials under 12 months >90% · MTTR-on-leak under 4 hours · Zombie findings = 0

PART 6

# About Cremit

Cremit is a Korean security company focused on the non-human identity problem. Our flagship product, **Argus**, is a secret detection and NHI governance platform used by engineering teams to find, attribute, and remediate credential exposure across source control, CI/CD, SaaS, and cloud.

Cremit Research publishes the NHI Kill Chain series, the NHI Exposure Severity Index, and ongoing field analysis of real-world credential exposure cases — much of it synthesized in this playbook.

## Ready to close your NHI gaps?

Try Argus free for 14 days. Connect your GitHub organization and get a ranked list of your ghost, shadow, aged, over-shared, and zombie keys within an hour. No credit card required.

Start free trial →