Engineer

設定

プログラムが特定の環境でどう動くかを決めるため、ソースコードの外から読み取る値や指定

どのようなものか

設定とは、ある環境でどう動くかを決めるために、プログラムが自分自身のソースコードの外から読み取るものすべてです。待ち受けるポート、接続するデータベース、利用を許すアカウント、有効にする機能などが該当します。英語の会話では、configurationをほとんどの場合configと略し、その情報を記すファイルをconfig fileと呼びます。NISTは、セキュリティを重視した構成管理の指針SP 800-128で、「情報システムまたはシステム構成要素を記述し、または構成するための、取り得る状態、パラメーター、仕様」と定義しています。ここで押さえるべき言葉は「記述」です。設定は状態を記述したものであり、システムがそれと一致しているとは限りません。

設定の取得元が一か所だけというプログラムは、ほとんどありません。既定値はバイナリーに組み込まれ、ディスク上の主設定ファイルの隣には、追加設定ファイルを置くディレクトリーがあり、起動時には環境変数やコマンドラインのフラグが渡されます。プログラムはこれらを一組の値にまとめます。同じキーを二か所で指定した場合の規則は、プログラムごとに異なります。OpenSSHのサーバーは、大半のキーワードについて最初に読んだ値を使いますが、後のファイルで先のファイルを上書きするツールもあります。同じファイルでも、読む順序が違えばシステムの動作は変わります。

形式も、INI形式の行、TOML、YAML、JSON、そのプログラム専用の形式など、さまざまです。TOML 1.0.0は「最小限の設定ファイル形式」を目指しています。一方、YAML 1.2.2は、設定ファイルにも使われる汎用的なデータ言語です。いずれにも、テキストを値に変換する独自の規則があります。

概念を判断に生かす

「設定とは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 構成全体をファイルに記述し、特定の人の記憶だけに頼らず、まったく同じ環境を再現できるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

設定によって、一度ビルドしたプログラムをさまざまな場所で使えます。同じプログラムをノートパソコン、ステージング用ホスト、本番で動かし、その違いを設定で表します。デプロイ可能なサービスの方法論であるTwelve-Factor Appは、ここを境界とします。設定とは、関連サービスへの接続情報、外部サービスの認証情報、各デプロイ先のホスト名など、「デプロイごとに変わる可能性があるすべて」です。設定とコードの厳密な分離を求め、どこでも試す価値のある判定を示しています。「認証情報を一切危険にさらさずに、コードベースをいつでもオープンソースとして公開できるか」という判定です。

同じ文書は、逆側の境界も示しています。ウェブフレームワークのルーティング表のように、すべてのデプロイで変わらない内部の結び付きは、その定義に含めず、「コードで扱うのが最適」としています。すべてのデプロイで同じ値は、設定の姿をしたコードです。それをファイルに置いても、間違いの起こる場所が一つ増えるだけです。

設定は、システムのセキュリティを表明する場所でもあります。誰がログインできるか、どのポートが応答するか、ファイアウォールが何を通すかを定めます。ホストの堅牢さは、実際に使っている値にかかっています。ファイルと実際の値の間には、パーサー、優先順位の規則、組み込みの既定値、プログラムのバージョン、環境があります。どれが変わっても、ファイルを変えずに結果が変わる可能性があります。したがって、確実な習慣は、プログラム自身に何を使っているかを尋ね、その答えを設定として扱うことです。

実際の活用例

このサイトは、一つのTOMLファイルhugo.tomlで設定されており、言語、タクソノミー、マウントなど、多くの項目を定めています。hugo.tomlにある一つのテーブルを見ると、数行でどれだけのことが決まるかが分かります。

```
[frontmatter]
  date = ["date", ":git", ":fileModTime"]
  lastmod = ["lastmod", ":git", ":fileModTime"]
```

各リストは検索順序です。まずページ自体に書かれた日付、次にそのファイルを変更した最後のコミットの日付、最後にファイルの更新日時を調べます。ファイルの上部近くにある `enableGitInfo = true` というフラグによって、Hugoがgitの履歴を読めるようになります。テーブルの上のコメントは、先頭の項目を説明しています。フロントマターのフィールド名そのものを指定しないと、ページに書いた `lastmod` は黙って無視され、そのまま通常どおりサイトがビルドされます。

このテーブルは、大半のページに同じ更新日時が付いたサイトマップの修正で説明している対策です。サイトマップは、サイトの176ページ中172ページが同じ瞬間に更新されたと検索エンジンに伝えていました。日付の取得元がファイルの更新日時だったため、すべてのファイルを書き直すコンテンツの書き出し処理を実行するたびに、全ファイルの時刻が一斉に付け直されていたのです。フォールバックの順序を変え、取得元をバージョン履歴に移しました。その記録された実装段階では、一か月分のコンテンツ編集後の再同期で変更されたのは、186ファイル中八つでした。

サーバーでも同じ教訓が、三段階の検証によるSSHの強化に表れています。自動化ロールは何か月もSSHデーモンの設定を書いていましたが、2026年9月のアクセス報告で、デーモンが最終的に採用した設定を読み取ると、二つの値がロールのファイルと食い違っていました。OS自身の既定値が追加設定ファイルより先に読まれ、その二項目では最初の指定が優先されていたのです。対策は、ベンダーのファイルより先に並ぶ接頭辞をファイル名に付けることでした。これに加えて、現在のロールは三段階で検証します。導入前に候補ファイルの構文を検査し、導入後に組み合わせた設定を検査し、最後に実効設定の出力を読み戻して24個のディレクティブを検証します。二つのディレクティブは、デーモンが実装しなくなったため、意図的に除外しています。

一度も機能していなかった三つのSSH総当たり攻撃対策は、同じ失敗を反対側から示します。2026年8月の調査で、各対策の設定を、稼働中のホストと照合しました。三つとも記述があり、毎回の報告にも表示されていましたが、ホストを構築した日から一つも役目を果たしていませんでした。jailの設定には、初期の強化でデーモンのポートを1986へ移したときに取り残された22番ポートが残っていました。レート制限自体の行は正しかったのですが、ファイアウォールのルール順序がそれを無効にしていました。より広い範囲を対象とするルールが上にあり、同じ通信を先に処理していたためです。遮断アクションの設定は、パッケージ管理システムが推奨するだけで必須にはしていないパケットフィルターの実行

ファイルを呼んでいました。このホストでは、その呼び出し先が一度も見つかっていませんでした。調査から二つの規則が生まれました。すべてのセキュリティ対策は、説明のコメントではなく、実際に試す検証で裏付けます。また、ファイアウォールはルールが一覧にあるかどうかではなく、各ルールがどの順序に置かれているかを確認します。

問題が起きるところ

以下はどれも、ファイルのテキストは正しいのに、実際の動作が誤っている例です。それぞれに、実際の動作を読み取る確認方法があります。

- 別の取得元が優先されます。OpenSSHのサーバー設定マニュアルは、「特に断りがない限り、各キーワードについて最初に得た値を使用します」と定めています。したがって、追加設定ファイルより先に読まれるベンダーの既定値は、追加設定に優先します。最後の値を使うツールなら、同じ配置でも結果は逆になります。プログラム自身のマニュアルで優先順位を読み、プログラムに確かめます。`sshd -t` は設定の妥当性を検査し、`sshd -T` は実効設定を表示します。
- TOMLのキーが別のテーブルに入ります。TOML 1.0.0では、`[table]` 見出しの下のキーと値の組は、「次の見出しまたはファイルの終端まで」そのテーブルに属し、字下げは無視されます。ファイルの末尾に加えたキーは、行頭に書いても最後のテーブルに入ります。テーブルは末尾に置き、テーブルに属さないキーはその上に置きます。キーやテーブルを二度定義することは無効であり、重複は上書きの方法ではなく、修正すべき誤りです。
- YAMLの引用符のない値が別の型になります。YAML 1.2.2が既定として推奨するコアスキーマでは、所定のパターンに一致する引用符のない値は、数値、真偽値、nullになります。`1.10` は浮動小数点数1.1、`0o17` は整数15、`0123` は123になり、`FALSE` は真偽値、`~` はnullです。文字列のままにすべき値にはすべて引用符を付けます。特に、バージョンを表す文字列と、先頭にゼロのある識別子が重要です。仕様は、マッピングのキーの重複も、読み込みが不完全な状態で終わる原因に挙げています。したがって、キーを二度書くことは欠陥です。
- 有効に見える設定が何もしません。先行するルールが優先されて適用されないルール、新しいリリースで廃止されたディレクティブ、サービスが既に使わなくなったポートは、いずれもファイルに残り、意図的な設定に見えます。22番ポートのjailも、より広いルールの下にあるレート制限も、正常と表示されていました。ファイルを個別に読むだけの検査は通ってしまいます。稼働中のシステムに対して、順序と存在を検証します。
- 秘密情報がファイルと一緒に移動します。設定ファイルのパスワードは、リポジトリ、バックアップ、不具合報告など、ファイルが行く先すべてへ付いていきます。Twelve-Factorが設定を環境変数に置く理由の一つは、設定ファイルと違い、「誤ってコードのリポジトリにコミットされる可能性が低い」ことです。

参考資料

- NIST CSRC用語集：設定 — NISTの各刊行物による定義を、出典文書とともに示します。
- TOML v1.0.0 — 一度に読み通せる長さの完全な仕様です。
- YAML Ain't Markup Language、バージョン1.2.2 — YAMLの仕様です。第10章に、引用符のない値の型を決めるコアスキーマがあります。
- Twelve-Factor App：設定 — デプロイごとに変わる値をコードの外に置く理由と、オープンソースとして公開できるかという判定を説明します。
- sshd_config(5) — 最初の値を使う規則、Include、Matchを含むOpenSSHサーバーの設定マニュアルです。
- sshd(8) — `-t` による設定の検査と、`-T` による実効設定の表示を説明するデーモンのマニュアルです。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「設定とは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/configuration/