Engineer

Konfiguration

Werte außerhalb des Codes, die das Programmverhalten steuern

Was es ist

Konfiguration ist alles, was ein Programm außerhalb seines eigenen Quellcodes liest, um sein Verhalten an einem bestimmten Ort festzulegen: den Port, auf dem es lauscht, die Datenbank, mit der es sich verbindet, zugelassene Konten und eingeschaltete Funktionen. Im Gespräch wird der Begriff fast immer zu Config verkürzt. Die Dateien dafür heißen Konfigurationsdateien. NIST definiert sie in SP 800-128, seinem Leitfaden für sicherheitsorientiertes Konfigurationsmanagement, als „die möglichen Zustände, Parameter und Spezifikationen, mit denen sich ein Informationssystem oder eine Systemkomponente beschreiben oder anordnen lässt“. Entscheidend ist beschreiben: Eine Konfiguration ist eine Beschreibung. Das System kann ihr entsprechen oder auch nicht.

Ein Programm bezieht seine Konfiguration selten von nur einem Ort. Voreinstellungen sind in die Binärdatei eingebaut. Eine Hauptdatei liegt auf der Platte, daneben ein Verzeichnis für ergänzende Drop-in-Fragmente. Beim Start kommen Umgebungsvariablen und Kommandozeilenoptionen hinzu. Das Programm führt sie zu einem Wertesatz zusammen. Was bei demselben Schlüssel in zwei Quellen geschieht, bestimmt jedes Programm selbst. Der OpenSSH-Server behält für die meisten Schlüsselwörter den zuerst gelesenen Wert. Andere Werkzeuge lassen eine spätere Datei eine frühere überschreiben. Dieselben Dateien ergeben in anderer Lesereihenfolge ein anderes System.

Die Formate unterscheiden sich: Zeilen im INI-Stil, TOML, YAML, JSON und programmspezifische Formate. TOML 1.0.0 will „ein minimales Konfigurationsdateiformat“ sein. YAML 1.2.2 ist eine allgemeine Sprache zur Darstellung von Daten, die auch Konfigurationsdateien nutzen. Jedes Format hat eigene Regeln, um Text in Werte umzuwandeln.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Konfiguration?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Durch Konfiguration kann ein Build an vielen Orten eingesetzt werden. Dasselbe Programm läuft auf einem Laptop, einem Staging-Host und in Produktion. Seine Konfiguration macht den Unterschied. Die Twelve-Factor App, eine Methodik für bereitstellbare Dienste, zieht dort die Grenze: Konfiguration ist „alles, was sich wahrscheinlich zwischen Bereitstellungen ändert“, etwa Verweise auf unterstützende Dienste, Zugangsdaten für externe Dienste und der Hostname jeder Bereitstellung. Sie verlangt strikte Trennung von Konfiguration und Code und bietet einen überall nützlichen Test: ob „die Codebasis jederzeit als Open Source veröffentlicht werden könnte, ohne Zugangsdaten zu gefährden“.

Derselbe Text zieht die andere Grenze. Interne Verknüpfungen, die bei jeder Bereitstellung gleich bleiben, etwa die Routingtabelle eines Webframeworks, gehören nicht zu seiner Definition und sind „am besten im Code aufgehoben“. Ein überall gleicher Wert ist als Einstellung verkleideter Code. Eine Datei gibt ihm nur einen weiteren Ort, an dem er falsch sein kann.

In der Konfiguration erklärt ein System auch seine Sicherheit: Wer sich anmelden darf, welche Ports antworten und was die Firewall durchlässt. Wie gut ein Host gehärtet ist, bestimmen die tatsächlich verwendeten Werte. Zwischen Datei und Werten liegen Parser, Vorrangregeln, einkompilierte Voreinstellungen, Programmversion und Umgebung. Jeder Faktor kann bei gleicher Datei ein anderes Ergebnis erzeugen. Verlässlich ist deshalb, das Programm nach seinen verwendeten Werten zu fragen und diese Antwort als Konfiguration zu behandeln.

Wie es in der Praxis aussieht

Diese Website wird über eine TOML-Datei, hugo.toml, konfiguriert. Sie setzt unter anderem Sprachen, Taxonomien und Mounts. Eine Tabelle darin zeigt, was wenige Zeilen entscheiden können:

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

Jede Liste ist eine Suchreihenfolge: ein in der Seite selbst geschriebenes Datum, dann das Datum des letzten Commits mit einer Änderung an der Datei, dann die Änderungszeit der Datei. Eine Option nahe dem Dateianfang, `enableGitInfo = true`, erlaubt Hugo überhaupt erst das Lesen der Git-Historie. Der Kommentar über der Tabelle erklärt den ersten Eintrag: Der Name des Front-Matter-Felds muss ausdrücklich in der Liste stehen. Sonst wird ein in einer Seite gesetztes `lastmod` stillschweigend ignoriert, während die Website wie gewohnt gebaut wird.

Diese Tabelle ist die Korrektur aus der Reparatur einer Sitemap mit einem gemeinsamen Zeitstempel für die meisten Seiten. Die Sitemap hatte Suchmaschinen gemeldet, dass 172 der damals 176 Seiten zum selben Zeitpunkt zuletzt geändert worden seien. Mit der Dateiänderungszeit als Datumsquelle stempelte jeder Inhaltsexport alle Dateien gleichzeitig neu ab, weil er jede davon erneut schreibt. Die Rückfallkette verlagerte die Quelle in die Versionshistorie. Im dokumentierten Implementierungsstadium betraf eine erneute Synchronisierung nach einem Monat mit Inhaltsänderungen acht von 186 Dateien.

Auf einem Server zeigt sich dieselbe Lehre bei der SSH-Härtung mit drei Validierungsstufen. Eine Automatisierungsrolle hatte seit Monaten die SSH-Daemon-Konfiguration geschrieben. Ein Zugriffsbericht im September 2026 las die aufgelösten Einstellungen des Daemons und fand zwei Abweichungen von der Rollendatei. Die eigenen Betriebssystemvorgaben standen vor dem Drop-in. Für diese beiden Einstellungen gewinnt das erste Vorkommen. Die Korrektur war deshalb ein Dateinamenpräfix, das vor dem des Herstellers sortiert. Die Rolle validiert jetzt in drei Stufen: Syntaxprüfung der Kandidatendatei vor der Installation, Prüfung der zusammengesetzten Konfiguration danach und

Zurücklesen der aufgelösten Ausgabe mit Assertions für 24 Direktiven. Zwei Direktiven blieben absichtlich außen vor, weil der Daemon sie nicht mehr implementiert.

Die drei nie wirksamen SSH-Schutzmaßnahmen gegen Brute Force zeigen die andere Seite. Im August 2026 legte ein Review die Konfiguration jeder Schutzmaßnahme neben den laufenden Host. Alle drei standen in den Dateien und jedem Bericht. Keine hatte seit dem Aufbau des Hosts ihre Aufgabe erfüllt. Die Jail-Konfiguration nannte weiter Port 22, übrig geblieben, als die erste Härtung den Daemon auf Port 1986 verlegte. Die Zeile der Ratenbegrenzung war korrekt. Die Firewallreihenfolge machte sie wirkungslos: Eine darüberstehende breitere Regel fing denselben Verkehr zuerst ab. Die Konfiguration der Sperraktion rief ein Paketfilterprogramm auf, das das Paketsystem empfiehlt, aber nicht voraussetzt. Auf diesem Host konnte dieser Befehl nie aufgelöst werden. Aus dem Review folgten zwei Regeln: Jede Sicherheitskontrolle wird durch eine Assertion abgesichert, die sie testet, niemals durch einen beschreibenden Kommentar. Eine Firewall wird anhand der Position jeder Regel geprüft, nicht danach, ob sie aufgelistet ist.

Wo es schiefgeht

In jedem folgenden Fall ist die Datei als Text korrekt, das tatsächliche Verhalten aber falsch. Für jeden Fall gibt es eine Prüfung, die das Verhalten ausliest.

- Die falsche Quelle gewinnt. Das OpenSSH-Handbuch zur Serverkonfiguration sagt: „Sofern nichts anderes angegeben ist, wird für jedes Schlüsselwort der zuerst erhaltene Wert verwendet.“ Eine vor dem eigenen Drop-in gelesene Herstellervorgabe gewinnt deshalb. Bei einem Werkzeug, dessen letzter Wert gewinnt, kehrt sich dasselbe Layout um. Lesen Sie die Vorrangregel im Handbuch des Programms und fragen Sie dann das Programm selbst: `sshd -t` prüft die Gültigkeit einer Konfiguration, `sshd -T` gibt die wirksame aus.
- Ein TOML-Schlüssel landet in der falschen Tabelle. Nach TOML 1.0.0 gehören die Paare unter einem `[table]` -Header „bis zum nächsten Header oder EOF“ zu dieser Tabelle. Einrückung wird ignoriert. Ein am Dateiende ergänzter Schlüssel gehört deshalb zur letzten Tabelle, selbst ganz links geschrieben. Platzieren Sie Tabellen am Ende und die nicht zu Tabellen gehörenden Schlüssel davor. Einen Schlüssel oder eine Tabelle zweimal zu definieren ist ungültig. Ein Duplikat ist deshalb ein zu behebender Fehler, niemals eine Überschreibmethode.

- Ein YAML-Wert ohne Anführungszeichen wechselt den Typ. Im Kernschema, das YAML 1.2.2 als Standard empfiehlt, wird ein nicht in Anführungszeichen gesetzter Wert bei passendem Muster zu Zahl, Boolean oder null: `1.10` ist die Gleitkommazahl 1,1, `0o17` die Ganzzahl 15, `0123` ergibt 123, `FALSE` ist ein Boolean und `~` ist null. Setzen Sie jeden als Text benötigten Wert in Anführungszeichen, zuerst Versionszeichenfolgen und Kennungen mit führenden Nullen. Die Spezifikation zählt auch wiederholte Mapping-Schlüssel zu den Ursachen unvollständig beendeter Ladevorgänge. Ein doppelt geschriebener Schlüssel ist deshalb ein Defekt.
- Eine Einstellung erscheint aktiv und tut nichts. Eine von einer früheren verdeckte Regel, eine in einer neueren Version entfernte Direktive, ein inzwischen vom Dienst verlassener Port: Alles bleibt in der Datei und wirkt absichtlich gesetzt. Die Jail auf Port 22 und die Ratenbegrenzung unter einer breiteren Regel wurden beide als funktionsfähig ausgegeben. Eine Prüfung einzelner Dateien lässt sie durch. Prüfen Sie deshalb Reihenfolge und Vorhandensein am laufenden System.
- Ein Geheimnis reist mit der Datei. Ein Passwort in einer Konfigurationsdatei geht überallhin mit: ins Repository, in Backups, in den Fehlerbericht. Twelve-Factor hält Konfiguration unter anderem deshalb in Umgebungsvariablen, weil im Gegensatz zu Konfigurationsdateien „kaum die Gefahr besteht, sie versehentlich ins Coderepository einzuchecken“.

Weiterführende Lektüre

- NIST-CSRC-Glossar: Konfiguration — die Definitionen aus NIST-Veröffentlichungen mit dem jeweiligen Quelldokument.
- TOML v1.0.0 — die vollständige Spezifikation, kurz genug, um sie am Stück zu lesen.
- YAML Ain’t Markup Language, Version 1.2.2 — die Spezifikation. Kapitel 10 enthält das Kernschema, das den Typ eines Werts ohne Anführungszeichen bestimmt.
- The Twelve-Factor App: Konfiguration — warum bereitstellungsspezifische Werte außerhalb des Codes bleiben sollen, samt Open-Source-Test.
- sshd_config(5) — das OpenSSH-Serverhandbuch mit Erstwertregel, Include und Match.
- sshd(8) — das Daemon-Handbuch, in dem `-t` die Konfiguration prüft und `-T` die wirksame ausgibt.



Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Konfiguration?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/configuration/