Engineer

SSH

別のコンピューター上で使う、通信を暗号化したシェル

どのようなものか

SSHはSecure Shellの略で、信頼できないネットワークを通じて別のコンピューターへログインし、コマンドを実行するためのネットワークプロトコルです。通信を暗号化し、先に両端がそれぞれの身元を証明します。この名前は三つのものを指します。一つは、IETFが2006年1月にRFC 4251を中心とする一連のRFCとして公開したプロトコルです。二つ目は、OpenBSDプロジェクトが保守する実装のOpenSSHで、この記事の例もこれを使います。三つ目は、`ssh` コマンドで、`sshd` というサーバーデーモンと通信します。

RFC 4251は、一つの接続の上に三つの層があると説明しています。通常は、IANAが割り当てたTCPポート22を使います。トランスポート層は鍵交換を行い、ホスト鍵でサーバーの身元を証明し、その後は通過するすべてを暗号化して完全性を検査します。ユーザー認証層はそのトンネル内で動き、クライアントが誰なのかを確定します。方法には、すべての実装が対応しなければならない公開鍵認証、対応が任意のパスワード認証、ホストに基づく信頼があります。接続層はトンネルをチャネルへ分割するため、一度のログインで、対話型シェル、一つのコマンド、SFTPによるファイル転送、複数のポート転送を同時に扱えます。

SSHは、セッションやパスワードを平文でネットワークへ送っていたtelnetとrloginに取って代わりました。現在使われているプロトコルはバージョン2です。

概念を判断に生かす

「SSH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。 アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

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

サーバーにとって、SSHは玄関です。データセンターやクラウドにあるマシンには、通常、手の届く場所にキーボードがありません。そのため、管理、デプロイ、ファイルのコピー、自動設定はすべてSSH経由で行われ、十分な権限でログインした人は、そのマシンを支配できます。

公開鍵認証は、SSHの使いやすさと強さを両立させるものです。利用者は秘密鍵を自分のマシンに保持し、対応する公開鍵をサーバーに置きます。既定の保存先は `~/.ssh/authorized_keys` です。ログイン時にはクライアントがそのセッションに結び付いたデータへ署名するので、秘密鍵そのものが回線を渡ることはありません。盗んだパスワードはどこからでも再利用でき、パスワードを受け付けるサーバーには、そのポートへ到達できる誰もが推測を試せます。自動化ツールは、まさにそれを行います。OpenSSHは初期状態ではパスワードを受け付けるため、無効化するかどうかはサーバーごとに決める必要があります。

信頼は逆方向にも必要です。クライアントも、正しいサーバーへ到達したことを知る必要があるからです。各サーバーはホスト鍵を持ち、RFC 4251は、その鍵を信頼する二つの方式を示しています。一つは、ホスト名と鍵を対応付ける一覧をクライアント上に置く方式です。もう一つは、認証局がホスト鍵へ署名し、クライアントはその認証局の鍵だけを保持する方式です。多くの人は一つ目を、最も緩い形で使っています。初回の接続では検証せずに鍵を受け入れ、`~/.ssh/known_hosts` に保存し、以後の接続では毎回それと照合します。RFCも、使いやすさのための妥協として、これを認めています。通常、trust on first useと呼ばれる方式です。最初のログインの安全性は、通過するネットワークの安全性に依存します。そのため、OpenSSHは新しい鍵を保存する前に確認し、既知のホストが別の鍵を提示すると接続を拒否します。

SSHは、ほかのツールの土台となる転送手段でもあります。Gitのプッシュとプル、 `scp` や `sftp` によるファイルのコピー、Ansibleによる管理対象ホストへの接続に使われます。ポート転送を使えば、サーバーのループバックインターフェースだけで待ち受けるデータベースへ、インターネットに公開せずノートパソコンから接続できます。

実際の活用例

当社のホストが対話型ログインを受け付ける手段はSSHだけです。設定を管理する自動化ロールを実行するたびに、強化設定を三段階で検証します。このチェックは2026年9月にさかのぼります。アクセスに関するレポートで、デーモンが最終的に採用する設定とロールが書き込んだ設定を比較したところ、二つが食い違っていました。ディストリビューションの既定値は、ロールのドロップインより先に読み込まれる有効な行として存在し、sshdは先の値を採用していたのです。ドロップインの名前を変更し、ベンダーのファイルより前に並ぶようにして、この差をなくしました。現在は、SSH設定が三つの検証を通る必要があります。配置前に候補ファイルの構文が正しく解析できること、配置後に組み合わせたファイル全体の検査を通ること、デーモンが最終的に採用する設定が期待する24個のディレクティブと一致することです。前にある行が優先され、書いた値が採用されなければ、実行は停止します。二つのディレクティブは意図的に対象から外しています。現在のデーモンが、どちらも廃止しているためです。

その一か月前、2026年8月のレビューでは、ホストを構築してから一度も機能していなかった三つの総当たり攻撃対策が見つかりました。デーモンが1986で待ち受けているのに、接続元を遮断するjailはポート22を監視していました。ファイアウォールのレート制限は、同じ通信を許可する、より広いルールの後にありました。最初に一致したルールが適用されるので、状態表示には掲載され続けていても、レート制限には一つのパケットも届きませんでした。遮断アクションは、パッケージマネージャーが推奨するだけで必須とはしないパケットフィルターツールに依存していました。そのため、jailは攻撃者を検出しても、遮断を試みるたびにエラーになっていました。このレビューから、二つの恒常的なルールが生まれました。すべてのセキュリティ制御には、実際に作用することを確認するチェックを付けること。そして、ファイアウォールのルールは、存在だけでなく位置も確認することです。

当社のGitフォージは、ログインする手段としてではなく、転送手段としてSSHを使います。SSHで操作でき、防御すべきウェブ管理画面を持たないことが、採用理由でした。匿名でも、鍵なしでも入れず、すべてのリポジトリは非公開です。SSHの待ち受けはループバック上のポート23231だけに限定し、外部の利用者は踏み台ホストを経由して到達します。そのため、フォージ用にファイアウォールのポートを追加で開放していません。

問題が起きるところ

強化設定の行がファイルにあっても、サーバーが無視している。 ほとんどのキーワードについて、 `sshd` は最初に読んだ値を採用します。 `Include` のファイルは辞書順に展開されるので、メインファイルが先にドロップインのディレクトリ

を読み込む場合、順序が先のベンダーのファイルが優先されても、両方のファイルは正しく見えます。設定を検索して分かるのは、その行を書いたことだけです。代わりにデーモンへ尋ねます。`sshd -t` は構文を検査し、`sshd -T` は実際に使う設定を表示します。

```
/etc/ssh/sshd_config.d/10-vendor.conf:     PasswordAuthentication yes
/etc/ssh/sshd_config.d/50-hardening.conf:  PasswordAuthentication no

$ sudo sshd -T | grep -i '^passwordauthentication'
passwordauthentication yes
```

変わったホスト鍵を削除し、受け入れてしまう。 既知のサーバーが別のホスト鍵を提示すると、`ssh` は警告を表示して接続を拒否します。反射的にホスト名を指定して `ssh-keygen -R` を実行し、再接続してyesと入力する行動は、中間者攻撃に必要な条件をすべて整えてしまいます。まず、別の経路で新しいフィンガープリントを確認します。例えば、サーバーのコンソールから、その公開ホスト鍵に対して `ssh-keygen -lf` を実行します。スクリプト内の `StrictHostKeyChecking no` は、見た目以上に危険です。新しい鍵を黙って保存するだけでなく、一定の制約はあるものの、鍵の変わったホストへの接続も許します。`accept-new` は未知のホストの鍵を保存しますが、変わった鍵は引き続き拒否します。

インターネット全体からパスワードログインを試せるままになる。 `PasswordAuthentication` の既定値はyesです。別の経路で同じパスワードを求められるkeyboard-interactive認証も、既定では有効です。全利用者に鍵が行き渡ったら、両方をnoに設定します。`PermitRootLogin` は既定ですでに `prohibit-password` なので、rootのパスワードは拒否しますが、rootの鍵は受け付けます。デーモンをポート22から移しても、変わるのは待ち受ける場所であって、誰が推測を試せるかではありません。前述のjailが示すとおり、そのポートを指定するすべての制御も、一緒に変更する必要があります。

エージェント転送が、サーバーへ鍵を貸してしまう。 `ForwardAgent` を有効にするか、`ssh -A` を使うと、rootを含め、遠隔ホストでファイル権限を回避できる人は誰でも、接続が続く間、あなたのエージェントを使って、別の場所であなたになりすまして認証を受けられます。鍵をコピーすることはできなくても、その鍵で署名できます。踏み台の背後にあるマシンへ到達するには、代わりにSSHのジャンプホスト機能を使います。`ssh -J bastion.example.com app.internal` は、踏み台を通してTCP接続を転送し、接続先との間でエンドツーエンドにセッションを確立します。踏み台は読めないバイト列を中継するだけで、鍵はノートパソコンに残ります。

参考資料

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — 三つの層、二つのホスト鍵の信頼モデル、セキュリティ上の考慮事項を説明しています。
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — 鍵交換、ホスト鍵の検証、双方が最初に送るバージョン文字列を説明しています。
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — 公開鍵、パスワード、ホストベースのログインが通信上でどう機能するかを説明しています。
- sshd_config(5) — サーバーの全キーワードとその既定値、最初に得た値を使うというルールを記載しています。
- ssh_config(5) — StrictHostKeyChecking、ProxyJump、ForwardAgentなど、クライアント側の設定を説明しています。
- sshd(8) — `-t` と `-T` の検査モード、authorized_keysとknown_hostsの書式を説明しています。

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

「SSH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/ssh/