Engineer

SSH

Verschlüsselte Shell auf einem anderen Rechner

Was es ist

SSH steht für Secure Shell und ist ein Netzwerkprotokoll, mit dem man sich auf einem anderen Rechner anmeldet und dort Befehle über ein nicht vertrauenswürdiges Netzwerk ausführt. Der Verkehr wird verschlüsselt, und beide Seiten weisen zuerst ihre Identität nach. Der Name bezeichnet drei Dinge: das Protokoll, das die IETF im Januar 2006 als eine Reihe von RFCs mit RFC 4251 als Ausgangspunkt veröffentlichte; OpenSSH, die vom OpenBSD-Projekt gepflegte Implementierung, auf die sich die Beispiele dieses Artikels beziehen; und den Befehl `ssh`, der mit einem Serverdaemon namens `sshd` spricht.

RFC 4251 beschreibt drei Schichten über einer Verbindung, normalerweise über TCP-Port 22, den die IANA dafür zugeteilt hat. Die Transportschicht führt den Schlüsselaustausch durch, weist die Identität des Servers mit dessen Hostschlüssel nach und verschlüsselt und prüft anschließend die Integrität aller übertragenen Daten. Innerhalb dieses Tunnels klärt die Benutzerauthentifizierung, wer der Client ist: über einen öffentlichen Schlüssel, was jede Implementierung unterstützen muss, über ein optionales Passwortverfahren oder über hostbasiertes Vertrauen. Die Verbindungsschicht teilt den Tunnel in Kanäle auf. Eine Anmeldung kann dadurch gleichzeitig eine interaktive Shell, einen einzelnen Befehl, eine SFTP-Dateiübertragung und mehrere weitergeleitete Ports transportieren.

SSH ersetzte telnet und rlogin, die Sitzungen und Passwörter als Klartext über das Netzwerk schickten. Heute wird Version 2 des Protokolls verwendet.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist SSH?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

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

Auf einem Server ist SSH die Eingangstür. Ein Rechner im Rechenzentrum oder in der Cloud hat selten eine für jemanden erreichbare Tastatur. Administration, Bereitstellungen, Dateikopien und automatisierte Konfiguration kommen deshalb über SSH an. Wer sich mit ausreichenden Rechten anmeldet, kontrolliert den Rechner.

Die Authentifizierung mit öffentlichen Schlüsseln macht SSH zugleich bequem und sicher. Der Benutzer behält einen privaten Schlüssel auf seinem eigenen Rechner und hinterlegt den zugehörigen öffentlichen Schlüssel auf dem Server, standardmäßig in `~/.ssh/authorized_keys`. Bei der Anmeldung signiert der Client Daten, die an diese Sitzung gebunden sind; der private Schlüssel selbst wird daher nie übertragen. Ein gestohlenes Passwort lässt sich von überall erneut verwenden. Jeder, der den Port eines Servers mit Passwortanmeldung erreicht, kann Passwörter durchprobieren; automatisierte Werkzeuge tun genau das. OpenSSH akzeptiert Passwörter im Auslieferungszustand. Sie abzuschalten ist deshalb eine Entscheidung, die für jeden Server getroffen werden muss.

Vertrauen wirkt auch in die andere Richtung: Der Client muss wissen, dass er den richtigen Server erreicht hat. Jeder Server besitzt einen Hostschlüssel. RFC 4251 nennt zwei Vertrauensmodelle: eine lokale Liste auf dem Client, die jedem Hostnamen seinen Schlüssel zuordnet, oder eine Zertifizierungsstelle, die Hostschlüssel signiert, sodass der Client nur den Schlüssel dieser Stelle vorhalten muss. Die meisten verwenden das erste Modell in seiner lockersten Form, die der RFC als Zugeständnis an die Benutzbarkeit zulässt: den Schlüssel beim ersten Kontakt ungeprüft übernehmen, in `~/.ssh/known_hosts` speichern und bei jeder späteren Verbindung damit vergleichen. Das wird meist Trust on First Use genannt, Vertrauen beim ersten Gebrauch. Die erste Anmeldung ist nur so sicher

wie das durchquerte Netzwerk. Deshalb fragt OpenSSH vor dem Speichern eines neuen Schlüssels nach und verweigert die Verbindung, wenn ein bekannter Host einen anderen vorlegt.

SSH ist außerdem ein Transport, auf dem andere Werkzeuge aufbauen. Git überträgt darüber Pushes und Pulls, `scp` und `sftp` kopieren Dateien, und Ansible erreicht darüber seine Hosts. Portweiterleitung erlaubt einem Laptop den Zugriff auf eine Datenbank, die nur an der Loopback-Schnittstelle eines Servers lauscht, ohne sie für das Internet zu öffnen.

Wie es in der Praxis aussieht

Der eigene Host des Unternehmens akzeptiert interaktive Anmeldungen ausschließlich über SSH. Seine Härtung wird bei jedem Lauf der für die Konfiguration zuständigen Automatisierungsrolle in drei Stufen geprüft. Die Prüfungen stammen vom September 2026. Damals verglich ein Zugangsbericht die vom Daemon ermittelten wirksamen Einstellungen mit den von der Rolle geschriebenen und fand zwei Abweichungen: Die Distributionsvorgaben waren aktive Konfigurationszeilen, die vor der Ergänzungsdatei der Rolle eingelesen wurden, und sshd behielt den zuerst gelesenen Wert. Die Ergänzungsdatei wurde so umbenannt, dass sie vor der Herstellerdatei einsortiert wird. Damit war die Lücke geschlossen. Heute muss eine SSH-Konfiguration drei Prüfungen bestehen: Der Entwurf wird vor der Installation auf gültige Syntax geprüft, die zusammengesetzten Dateien werden nach der Installation geprüft, und die ermittelten wirksamen Einstellungen des Daemons werden mit 24 erwarteten Direktiven verglichen. Ein Wert, der gegenüber einer früheren Zeile verliert, stoppt damit den Lauf. Zwei Direktiven sind bewusst ausgenommen: Der aktuelle Daemon unterstützt beide nicht mehr.

Einen Monat zuvor hatte eine Prüfung im August 2026 drei Schutzmaßnahmen gegen Brute-Force-Angriffe gefunden, die seit dem Aufbau des Hosts nie funktioniert hatten. Das Sperr-Jail überwachte Port 22, während der Daemon auf 1986 lauschte. Die Ratenbegrenzung der Firewall stand hinter einer allgemeineren Regel, die denselben Verkehr erlaubte. Da die erste passende Regel gewinnt, kam nie ein Paket bei der Begrenzung an, obwohl die Statusausgabe sie weiter aufführte. Die Sperraktion war auf ein Paketfilter-Werkzeug angewiesen, das der Paketmanager nur empfiehlt. Das Jail erkannte deshalb Angreifer und scheiterte dann bei jedem Sperrversuch. Aus der Prüfung ergaben sich zwei feste Regeln: Zu jeder Sicherheitskontrolle gehört eine Prüfung, die ihre tatsächliche Wirkung nachweist, und eine Firewallregel wird anhand ihrer Position ebenso wie anhand ihrer Anwesenheit bestätigt.

Die eigene Git-Forge des Unternehmens nutzt SSH als Transport statt als Anmeldeweg. Sie wurde gewählt, weil sie sich über SSH bedienen lässt und kein Web-Administrationspanel abgesichert werden muss. Niemand erhält anonym oder ohne Schlüssel Zugang, und jedes Repository ist privat. Ihr SSH-Dienst akzeptiert Verbindungen ausschließlich auf Loopback an Port 23231. Von außen erreicht man ihn über einen Sprunghost, weshalb die Firewall dafür keinen zusätzlichen Port öffnet.

Wo es schiefgeht

Eine Härtungseinstellung steht in der Datei und wird ignoriert. Für die meisten Schlüsselwörter behält `sshd` den ersten gelesenen Wert. Ein `Include` bindet seine Dateien in lexikografischer Reihenfolge ein. Liest die Hauptdatei zuerst ein Verzeichnis mit Ergänzungsdateien, gewinnt daher eine früh einsortierte Herstellerdatei gegen die eigene, obwohl beide Dateien richtig aussehen. Eine Suche in der Konfiguration beweist nur, dass eine Zeile geschrieben wurde. Fragen Sie stattdessen den Daemon: `sshd -t` prüft die Syntax, `sshd -T` gibt die tatsächlich verwendeten Einstellungen aus.

```
/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
```

Ein geänderter Hostschlüssel wird gelöscht und akzeptiert. Legt ein bekannter Server einen anderen Hostschlüssel vor, warnt `ssh` und verweigert die Verbindung. Der Reflex besteht darin, `ssh-keygen -R` für den Hostnamen auszuführen, sich erneut zu verbinden und yes einzugeben. Mehr braucht ein Man-in-the-Middle nicht. Bestätigen Sie den neuen Fingerabdruck zuerst über einen anderen Kanal, etwa mit `ssh-keygen -lf` für den öffentlichen Hostschlüssel des Servers an dessen Konsole. In Skripten ist `StrictHostKeyChecking no` gefährlicher, als es aussieht: Es speichert nicht nur neue Schlüssel stillschweigend, sondern lässt innerhalb gewisser Grenzen auch Verbindungen zu Hosts mit geänderten Schlüsseln zu. Die Einstellung `accept-new` speichert unbekannte Hosts und lehnt geänderte Schlüssel weiterhin ab.

Passwortanmeldungen bleiben aus dem ganzen Internet möglich. `PasswordAuthentication` steht standardmäßig auf yes, ebenso die tastaturinteraktive Authentifizierung, die dasselbe Passwort auf einem anderen

Weg abfragen kann. Sobald alle Benutzer einen Schlüssel haben, setzen Sie beides auf no. `PermitRootLogin` steht bereits standardmäßig auf `prohibit-password` : Passwörter für root sind damit verboten, ein Root-Schlüssel bleibt aber zulässig. Den Daemon auf einen anderen Port als 22 umzustellen ändert, wo er lauscht, nicht aber, wer Passwörter durchprobieren kann. Jede Kontrolle mit einer Portangabe muss mitziehen, wie das obige Sperr-Jail zeigt.

Agent-Weiterleitung leiht dem Server Ihre Schlüssel. Mit aktiviertem `ForwardAgent` oder `ssh -A` kann jeder, der Dateiberechtigungen auf dem entfernten Host umgehen kann, einschließlich root, Ihren Agenten nutzen, um sich während Ihrer Verbindung andernorts als Sie anzumelden. Den Schlüssel kopieren kann diese Person nicht, damit signieren aber schon. Für einen Rechner hinter einem Bastion-Host verwenden Sie stattdessen einen Sprunghost: `ssh -J bastion.example.com app.internal` leitet eine TCP-Verbindung durch die Bastion und handelt die Sitzung mit dem Ziel Ende zu Ende aus. Die Bastion reicht damit nur Bytes weiter, die sie nicht lesen kann, und der Schlüssel bleibt auf Ihrem Laptop.

Weiterführende Lektüre

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — die drei Schichten, die beiden Vertrauensmodelle für Hostschlüssel und die Sicherheitsüberlegungen.
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — Schlüsselaustausch, Hostschlüsselprüfung und die zuerst von beiden Seiten gesendete Versionszeichenfolge.
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — wie Anmeldungen per öffentlichem Schlüssel, Passwort und hostbasiertem Vertrauen auf dem Übertragungsweg funktionieren.
- sshd_config(5) — alle Serverschlüsselwörter mit ihren Standardwerten und die Regel, den zuerst ermittelten Wert zu verwenden.
- ssh_config(5) — die Clientseite, einschließlich StrictHostKeyChecking, ProxyJump und ForwardAgent.
- sshd(8) — die Prüfmodi `-t` und `-T` sowie die Formate von authorized_keys und known_hosts.

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 SSH?“ 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/ssh/