Engineer

SSH

ett krypterat skal på en annan maskin

Vad det är

SSH, förkortning av Secure Shell, är ett nätverksprotokoll för att logga in på en annan dator och köra kommandon på den över ett nätverk som ingen litar på, med trafiken krypterad och med båda ändarna som först bevisar vilka de är. Namnet täcker tre saker: protokollet, som IETF publicerade i januari 2006 som en serie RFC:er med RFC 4251 i spetsen; OpenSSH, den implementation som OpenBSD-projektet underhåller och som exemplen i den här artikeln använder; och kommandot `ssh`, som talar med en serverdaemon som heter `sshd`.

RFC 4251 beskriver tre lager över en och samma anslutning, normalt TCP-port 22, det nummer som IANA har tilldelat protokollet. Transportlagret sköter nyckelutbytet, bevisar serverns identitet med dess värdnyckel och krypterar och integritetskontrollerar sedan allt som passerar. Lagret för autentisering av användare körs inuti den tunneln och avgör vem klienten är: med publik nyckel, som varje implementation måste stödja, med lösenord, som är valfritt, eller med förtroende baserat på värddatorn. Anslutningslagret delar upp tunneln i kanaler, så en inloggning kan bära ett interaktivt skal, ett enskilt kommando, en filöverföring med SFTP och flera vidarebefordrade portar på samma gång.

SSH ersatte telnet och rlogin, som skickade sessioner och lösenord över nätverket i klartext. Det protokoll som används i dag är version 2.

Använd begreppet i ett beslut

Använd ”Vad är SSH?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Konton, lösenord och behörigheter ordnade så att alla har exakt den åtkomst deras arbete kräver, och de som slutar förlorar den samma dag.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Skrev 1 000 RBAC-regler som säkrar Linux-servrar — 1,000 åtkomstregler, skrivna för att täppa till en säkerhetslucka

Skrev 1 000 RBAC-regler som säkrar Linux-servrar Hör av er →

Varför det spelar roll

På en server är SSH ytterdörren. En maskin i ett datacenter eller i molnet har sällan ett tangentbord inom räckhåll för någon, så administration, driftsättningar, filkopior och automatiserad konfiguration kommer alla in genom SSH, och den som loggar in med tillräckliga privilegier äger maskinen.

Autentisering med publik nyckel är det som låter SSH vara bekvämt och starkt på samma gång. Användaren behåller en privat nyckel på sin egen maskin och lägger den matchande publika nyckeln på servern, som standard i `~/.ssh/authorized_keys`. Vid inloggningen signerar klienten data som är knutna till just den sessionen, så själva nyckeln skickas aldrig över nätverket. Ett stulet lösenord kan spelas upp igen var som helst ifrån, och en server som tar emot lösenord kan gissas mot av vem som helst som når dess port; automatiserade verktyg gör precis det. OpenSSH tar emot lösenord från start, så att stänga av dem är ett beslut som varje server måste fatta.

Förtroendet går också åt andra hållet, eftersom klienten behöver veta att den nådde rätt server. Varje server har en värdnyckel, och RFC 4251 erbjuder två sätt att lita på den: en lokal lista hos klienten som kopplar varje värdnamn till dess nyckel, eller en certifikatutfärdare som signerar värdnycklar så att klienten bara behöver utfärdarens nyckel. De flesta använder det första i dess lösaste form, som RFC:n godtar som en eftergift åt användbarheten: ta emot nyckeln okontrollerad vid första kontakten, spara den i `~/.ssh/known_hosts` och jämför mot den vid varje senare anslutning. Det brukar kallas förtroende vid första användning (trust on first use). Den första inloggningen är bara så säker som nätverket den passerar, och därför frågar OpenSSH innan den sparar en ny nyckel och vägrar rakt av när en känd värd visar upp en annan.

SSH är också en transport som andra verktyg bygger på. Git skickar och hämtar över det, `scp` och `sftp` kopierar filer med det, Ansible når sina värdar genom det, och vidarebefordran av portar låter en bärbar dator nå en databas som bara lyssnar på en servers loopback-gränssnitt, utan att öppna den mot internet.

Så ser det ut i praktiken

Företagets egen värd tar emot interaktiva inloggningar över SSH och inget annat, och dess härdning kontrolleras i tre steg varje gång automatiseringsrollen som äger konfigurationen körs. Kontrollerna är från september 2026, när en åtkomstrapport jämförde de inställningar som daemonen hade kommit fram till med dem som rollen skrev, och två skilde sig åt: distributionens standardvärden var aktiva rader som lästes före rollens drop-in-fil, och sshd behöll det tidigare värdet. Att byta namn på drop-in-filen så att den sorteras före leverantörens fil stängde glappet. Nu syntaxkontrolleras den nya filen innan den installeras, hela konfigurationen kontrolleras när den är på plats, och daemonens gällande inställningar jämförs med 24 förväntade direktiv, så att ett värde som förlorar mot en tidigare rad stoppar körningen. Två direktiv utelämnas medvetet: daemonen i sin nuvarande version har tagit bort båda.

En månad tidigare, i augusti 2026, hittade en granskning tre skydd mot brute force-attacker som aldrig hade fungerat sedan värden byggdes. Jailen för spärrning bevakade port 22 medan daemonen lyssnade på 1986. Brandväggens hastighetsbegränsning kom efter en bredare regel som tillät samma trafik, och eftersom den första matchande regeln vinner såg begränsningen aldrig ett enda paket, fast statusutskriften fortsatte att lista den. Spärråtgärden förlitade sig på ett verktyg för paketfiltrering som pakethanteraren bara rekommenderar, så jailen upptäckte angripare och gav sedan ett fel varje gång den försökte spärra någon. Granskningen lämnade två bestående regler efter sig: varje säkerhetsskydd backas upp av en kontroll som bevisar att det verkar, och en regel i brandväggen bekräftas utifrån sin position och inte bara utifrån att den finns.

Företagets egen git-forge använder SSH som transport snarare än som ett sätt att logga in. Den valdes för att den styrs över SSH och inte har någon administrativ webbpanel att försvara. Ingen kommer in anonymt eller utan nyckel, och varje repo är privat. Dess SSH-lyssnare tar bara emot anslutningar på loopback, på port 23231, och de som är utanför når den genom att gå via en hoppserver, så brandväggen öppnar ingen extra port för den.

Var det går fel

En härdningsrad står i filen, och servern ignorerar den. För de flesta nyckelord behåller `sshd` det första värde den läser, och en `Include` expanderar sina filer i lexikal ordning, så när huvudfilen drar in en katalog med drop-in-filer först vinner en leverantörsfil som sorteras tidigt över din, och båda filerna ser ändå rätt ut.

Att söka i konfigurationen bevisar bara att en rad skrevs. Fråga daemonen i stället: `sshd -t` kontrollerar syntaxen, och `sshd -T` skriver ut de inställningar den faktiskt kommer att använda.

```
/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
```

En ändrad värdnyckel raderas och godtas. När en känd server visar upp en annan värdnyckel skriver `ssh` ut en varning och vägrar ansluta. Reflexen är `ssh-keygen -R` på värdnamnet, en ny anslutning och ett inskrivet yes, vilket är allt en mellanhand i en man-in-the-middle-attack behöver. Bekräfta det nya fingeravtrycket via en annan kanal först, t.ex. med `ssh-keygen -lf` på serverns publika värdnyckel från dess konsol. I skript är `StrictHostKeyChecking no` värre än det ser ut: utöver att spara nya nycklar i tysthet låter det, inom vissa gränser, anslutningar till värdar vars nycklar har ändrats gå vidare. Inställningen `accept-new` sparar okända värdar och vägrar ändå ändrade nycklar.

Inloggning med lösenord står öppen mot hela internet. `PasswordAuthentication` är som standard yes, och det är även autentisering med tangentbordsinteraktion, som kan fråga efter samma lösenord en annan väg. När varje användare har en nyckel sätter du båda till no. `PermitRootLogin` har redan `prohibit-password` som standard, vilket vägrar lösenord för root men ändå släpper in en nyckel för root. Att flytta daemonen från port 22 ändrar var den lyssnar, inte vem som kan gissa mot den, och varje skydd som namnger porten måste flyttas med, som jailen ovan visar.

Vidarebefordran av agenten lånar ut dina nycklar till servern. Med `ForwardAgent` påslaget, eller `ssh -A` , kan den som kan kringgå filbehörigheterna på fjärrvärden, root inräknad, använda din agent för att autentisera sig någon annanstans som du så länge du är ansluten. Nyckeln går inte att kopiera, men den går att signera med. För att nå en maskin bakom en bastion använder du en hoppserver i stället: `ssh -J bastion.example.com app.internal` vidarebefordrar en TCP-anslutning genom bastionen och förhandlar sessionen med målet från ände till ände, så bastionen förmedlar byte som den inte kan läsa och nyckeln stannar på din bärbara dator.

Läs mer

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — de tre lagren, de två modellerna för förtroende för värdnycklar och säkerhetsövervägandena.
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — nyckelutbyte, verifiering av värdnyckeln och den versionssträng som varje sida skickar först.
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — hur inloggning med publik nyckel, lösenord och värdbaserat förtroende fungerar på nätverksnivå.
- sshd_config(5) — varje nyckelord på serversidan med sitt standardvärde, och regeln att det första värde som hittas används.
- ssh_config(5) — klientsidan, bl.a. StrictHostKeyChecking, ProxyJump och ForwardAgent.
- sshd(8) — testlägena `-t` och `-T` och formaten för authorized_keys och known_hosts.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är SSH?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/ssh/