Engineer

SSH

en krypteret skal på en anden maskine

Hvad det er

SSH, kort for Secure Shell, er en netværksprotokol til at logge ind på en anden computer og køre kommandoer på den over et netværk, man ikke kan stole på, med trafikken krypteret og begge ender, der beviser, hvem de er, før arbejdet begynder. Navnet dækker tre ting: protokollen, som IETF udgav i januar 2006 som en række RFC'er med RFC 4251 i spidsen; OpenSSH, den implementering, som OpenBSD-projektet vedligeholder, og som eksemplerne her bruger; og selve kommandoen `ssh`, der taler med en serverdæmon ved navn `sshd`.

RFC 4251 beskriver tre lag oven på én forbindelse, normalt TCP til port 22, det nummer IANA har tildelt protokollen. Transportlaget klarer nøgleudvekslingen, beviser serverens identitet med dens værtsnøgle og krypterer og integritetstjekker derefter alt, der passerer. Laget for brugerautentificering kører inde i den tunnel og afgør, hvem klienten er: med offentlig nøgle, som alle implementeringer skal understøtte, med adgangskode, som er valgfri, eller med værtsbaseret tillid. Forbindelseslaget deler tunnelen op i kanaler, så ét login kan bære en interaktiv skal, en enkelt kommando, en SFTP-filoverførsel og flere viderestillede porte på samme tid.

SSH afløste telnet og rlogin, der sendte sessioner og adgangskoder over netværket som klartekst. Den protokol, der bruges i dag, er version 2.

Brug begrebet i en beslutning

Brug »Hvad er SSH?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Konti, adgangskoder og rettigheder sat op, så folk har præcis den adgang, jobbet kræver, og fratrådte mister den samme dag.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

På en server er SSH hoveddøren. En maskine i et datacenter eller en sky har sjældent et tastatur inden for rækkevidde, så administration, udrulninger, filkopier og automatiseret konfiguration kommer alle ind via SSH, og den, der logger ind med tilstrækkelige rettigheder, ejer maskinen.

Autentificering med offentlig nøgle er det, der gør SSH både bekvem og stærk. Brugeren beholder en privat nøgle på sin egen maskine og lægger den tilhørende offentlige nøgle på serveren, som udgangspunkt i `~/.ssh/authorized_keys`. Ved login signerer klienten data, der er bundet til netop den session, så selve nøglen aldrig sendes over nettet. En stjålet adgangskode kan genbruges hvor som helst fra, og en server, der tager imod adgangskoder, kan udsættes for gætteri fra alle, der når frem til dens port; automatiserede værktøjer gør præcis det. OpenSSH accepterer adgangskoder fra fabrikken, så det er en beslutning for hver enkelt server at slå dem fra.

Tilliden skal også gå den anden vej, for klienten skal vide, at den har ramt den rigtige server. Hver server har en værtsnøgle, og RFC 4251 giver to måder at stole på den: en lokal liste hos klienten, der knytter hvert værtsnavn til dets nøgle, eller en certifikatudsteder, der signerer værtsnøgler, så klienten kun behøver udstederens nøgle. De fleste bruger den første i dens løseste form, som RFC'en accepterer af hensyn til brugervenligheden: tag nøglen ukontrolleret ved første kontakt, gem den i `~/.ssh/known_hosts`, og sammenlign med den ved hver senere forbindelse. Det kaldes som regel trust on first use, tillid ved første brug. Det første login er kun så sikkert som det netværk, det krydser, og derfor spørger OpenSSH, før den gemmer en ny nøgle, og afviser blankt, når en kendt vært viser en anden.

SSH er også en transport, som andre værktøjer bygger på. Git skubber og henter over den, `scp` og `sftp` kopierer filer med den, Ansible når sine værter gennem den, og viderestilling af porte lader en bærbar nå en database, der kun lytter på serverens loopback-interface, uden at databasen åbnes mod internettet.

Sådan ser det ud i praksis

Virksomhedens egen vært tager kun imod interaktive logins via SSH, og dens hærdning tjekkes i tre trin, hver gang den automatiseringsrolle, der ejer konfigurationen, kører. Tjekkene stammer fra september 2026, da en adgangsrapport sammenlignede de indstillinger, dæmonen havde slået fast, med dem, rollen skrev, og to var forskellige: distributionens standardværdier var aktive linjer, der blev læst før rollens drop-in-fil, og sshd beholdt den tidligste værdi. Et nyt filnavn, der sorterer før leverandørens fil, lukkede hullet. Nu syntakstjekkes den nye fil, før den installeres, hele konfigurationen tjekkes, når den er på plads, og dæmonens endelige indstillinger sammenlignes med 24 forventede direktiver, så en værdi, der taber til en tidligere linje, stopper kørslen. To direktiver er udeladt med vilje, fordi den nuværende dæmon har droppet dem begge.

En måned tidligere fandt en gennemgang tre beskyttelser mod brute force, der aldrig havde virket, heller ikke da værten var ny. Gennemgangen fandt sted i august 2026. Jailen til blokering holdt øje med port 22, mens dæmonen lyttede på 1986. Firewallens hastighedsgrænse stod efter en bredere regel, der tillod den samme trafik, og da den første regel, der passer, vinder, nåede trafikken aldrig frem til grænsen, selvom statusoversigten blev ved med at vise den. Blokeringshandlingen afhang af et pakkefilterværktøj, som pakkesystemet kun anbefaler, så jailen opdagede angriberne og fejlede derefter, hver gang det forsøgte at blokere en af dem. Gennemgangen efterlod to faste regler: hver sikkerhedskontrol skal bakkes op af et tjek, der viser, at den virker, og en firewallregel bekræftes på sin plads i rækkefølgen og ikke kun på, at den findes.

Virksomhedens egen git-forge bruger SSH som transport og ikke som en måde at logge ind på. Den blev valgt, fordi den styres over SSH og intet webpanel har, der skal forsvares. Anonyme brugere og brugere uden nøgle bliver afvist, og alle repositorier er private. Dens SSH-lytter tager kun imod forbindelser på loopback, på port 23231, og folk udefra når den ved at hoppe gennem en jump host, så firewallen åbner ingen ekstra port for den.

Hvor det går galt

En hærdningslinje står i filen, og serveren ignorerer den. For de fleste nøgleord beholder `sshd` den første værdi, den læser, og et `Include` folder sine filer ud i leksikalsk rækkefølge, så når hovedfilen trækker en mappe med drop-in-filer ind først, slår en leverandørfil, der sorterer tidligt, din egen, og begge filer

ser stadig rigtige ud. At søge i konfigurationen beviser kun, at en linje er skrevet. Spørg dæmonen i stedet: `sshd -t` tjekker syntaksen, og `sshd -T` udskriver de indstillinger, den faktisk vil bruge.

```
/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 ændret værtsnøgle bliver slettet og accepteret. Når en kendt server viser en anden værtsnøgle, skriver `ssh` en advarsel og nægter at forbinde. Refleksen er `ssh-keygen -R` på værtsnavnet, en ny forbindelse og et indtastet yes, og det er alt, en mand i midten har brug for. Bekræft det nye fingeraftryk ad en anden kanal først, for eksempel med `ssh-keygen -lf` på serverens offentlige værtsnøgle fra dens konsol. I scripts er `StrictHostKeyChecking no` værre, end det ser ud: ud over at gemme nye nøgler i stilhed lader det forbindelser til værter med ændrede nøgler fortsætte, inden for visse grænser. Indstillingen `accept-new` gemmer ukendte værter og afviser stadig ændrede nøgler.

Logins med adgangskode står åbne for hele internettet. `PasswordAuthentication` er som udgangspunkt yes, og det samme gælder keyboard-interactive-autentificering, der kan bede om den samme adgangskode ad en anden vej. Når alle brugere har en nøgle, så sæt begge til no. `PermitRootLogin` står allerede som udgangspunkt på `prohibit-password`, der afviser adgangskoder for root, men stadig lukker en root-nøgle ind. At flytte dæmonen væk fra port 22 ændrer, hvor den lytter, men ikke, hvem der kan gætte løs, og hver kontrol, der nævner porten, skal flytte med, som jailen ovenfor viser.

Agent forwarding låner dine nøgler ud til serveren. Med `ForwardAgent` slået til, eller `ssh -A`, kan enhver, der kan omgå filrettighederne på den fjerne vært, root medregnet, bruge din agent til at logge ind andre steder som dig, så længe du er forbundet. Nøglen kan de ikke kopiere, men de kan signere med den. Brug i stedet en jump host for at nå en maskine bag en bastion: `ssh -J bastion.example.com app.internal` viderestiller en TCP-forbindelse gennem bastionen og forhandler sessionen med målet fra ende til ende, så bastionen videresender bytes, den ikke kan læse, og nøglen bliver på din bærbare.

Læs mere

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — de tre lag, de to tillidsmodeller for værtsnøgler og sikkerhedsovervejelserne.
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — nøgleudveksling, kontrol af værtsnøglen og den versionsstreng, hver side sender først.
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — hvordan logins med offentlig nøgle, adgangskode og værtsbaseret tillid foregår på linjen.
- sshd_config(5) — alle serverens nøgleord med standardværdier og reglen om, at den første værdi, der hentes, er den, der bruges.
- ssh_config(5) — klientsiden, herunder StrictHostKeyChecking, ProxyJump og ForwardAgent.
- sshd(8) — testtilstandene `-t` og `-T` og formaterne for authorized_keys og known_hosts.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er SSH?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/ssh/