Engineer

SSH

et kryptert skall på en annen maskin

Hva det er

SSH, kort for Secure Shell, er en nettverksprotokoll for å logge inn på en annen datamaskin og kjøre kommandoer der over et nettverk ingen stoler på, med trafikken kryptert og med begge ender som beviser hvem de er, før noe annet skjer. Navnet dekker tre ting. Det er protokollen, som IETF publiserte i januar 2006 som en serie RFC-er med RFC 4251 i spissen. Det er OpenSSH, implementasjonen OpenBSD-prosjektet vedlikeholder, og den eksemplene i denne artikkelen bruker. Og det er kommandoen `ssh`, som snakker med en serverdaemon som heter `sshd`.

RFC 4251 beskriver tre lag over én tilkobling, normalt TCP-port 22, nummeret IANA har tildelt protokollen. Transportlaget kjører nøkkelutvekslingen, beviser serverens identitet med vertsnøkkelen og krypterer og integritetssjekker deretter alt som går over. Laget for brukerautentisering kjører inne i den tunnelen og avgjør hvem klienten er: med offentlig nøkkel, som hver implementasjon må støtte, med passord, som er valgfritt, eller med tillit basert på verten. Tilkoblingslaget deler tunnelen opp i kanaler, så én innlogging kan bære et interaktivt skall, en enkelt kommando, en filoverføring med SFTP og flere videresendte porter samtidig.

SSH avløste telnet og rlogin, som sendte både økter og passord over nettverket som ren tekst. Protokollen som brukes i dag, er versjon 2.

Bruk begrepet i en beslutning

Bruk «Hva er SSH?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Kontoer, passord og tilganger ordnet slik at folk har nøyaktig den tilgangen jobben krever, og de som slutter, mister den samme dag.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Skrev 1 000 RBAC-regler for å sikre Linux-servere — 1,000 tilgangsregler skrevet for å tette et sikkerhetshull

Skrev 1 000 RBAC-regler for å sikre Linux-servere Ta kontakt →

Hvorfor det er viktig

På en server er SSH inngangsdøren. En maskin i et datasenter eller i en sky har sjelden et tastatur innen rekkevidde for noen, så administrasjon, produksjonssettinger, filkopier og automatisert konfigurasjon kommer alle inn gjennom SSH, og den som logger inn med høye nok rettigheter, eier maskinen.

Autentisering med offentlig nøkkel er det som lar SSH være både bekvemt og sterkt. Brukeren holder en privat nøkkel på sin egen maskin og legger den tilhørende offentlige nøkkelen på serveren, som standard i `~/.ssh/authorized_keys`. Ved innlogging signerer klienten data knyttet til den aktuelle økten, så selve nøkkelen krysser aldri ledningen. Et stjålet passord kan spilles av igjen fra hvor som helst, og en server som godtar passord, kan utsettes for gjetting fra alle som når porten, noe automatiserte verktøy gjør hele tiden. OpenSSH godtar passord rett fra esken, så det å slå dem av er en beslutning hver server må ta.

Tilliten går også den andre veien, fordi klienten må vite at den har nådd den riktige serveren. Hver server har en vertsnøkkel, og RFC 4251 gir to måter å stole på den. Den ene er en lokal liste hos klienten som knytter hvert vertsnavn til nøkkelen sin. Den andre er en sertifiseringsinstans som signerer vertsnøkler, slik at klienten bare trenger instansens nøkkel. De fleste bruker den første i sin løseste form, som RFC-en godtar som en innrømmelse til brukervennligheten: ta imot nøkkelen usjekket ved første kontakt, lagre den i `~/.ssh/known_hosts` og sammenlign med den ved hver senere tilkobling. Dette kalles gjerne tillit ved første bruk. Den første innloggingen er bare så trygg som nettverket den krysser, og derfor spør OpenSSH før den lagrer en ny nøkkel, og nekter blankt når en kjent vert viser fram en annen.

SSH er dessuten en transport andre verktøy bygger på. Git dytter og henter over den, `scp` og `sftp` kopierer filer med den, Ansible når vertene sine gjennom den, og videresending av porter lar en bærbar maskin nå en database som bare lytter på loopback-grensesnittet til en server, uten å åpne den mot internett.

Slik ser det ut i praksis

Selskapets egen vert godtar interaktive innlogginger over SSH og ingenting annet, og herdingen sjekkes i tre trinn hver gang automatiseringsrollen som eier konfigurasjonen, kjører. Sjekkene er fra september 2026, da en tilgangsrapport sammenlignet innstillingene daemonen faktisk hadde kommet frem til, med dem rollen skrev, og to av dem var forskjellige. Distribusjonens standardverdier var aktive linjer som ble lest før rollens drop-in-fil, og `sshd` beholdt den tidligste verdien. Å gi drop-in-filen et navn som sorteres foran leverandørens fil, tettet hullet. Nå syntakssjekkes den nye filen før den installeres, hele konfigurasjonen sjekkes når den er på plass, og de endelige innstillingene til daemonen sammenlignes med 24 forventede direktiver, så en verdi som taper mot en tidligere linje, stopper kjøringen. To direktiver er utelatt med vilje: den gjeldende daemonen har fjernet begge.

En måned tidligere, i august 2026, fant en gjennomgang tre beskyttelser mot brute force som aldri hadde virket siden verten ble bygget. Utestengingsjailen overvåket port 22 mens daemonen lyttet på 1986. Ratebegrensningen i brannmuren kom etter en bredere regel som tillot den samme trafikken, og siden den første regelen som treffer, vinner, så begrensningen aldri en eneste pakke, selv om statusoversikten fortsatte å vise den. Utestengingshandlingen var avhengig av et pakkefilterverktøy som pakkesystemet bare anbefaler, så jailen fant angriperne og feilet deretter hver gang den prøvde å stenge en av dem ute. Gjennomgangen etterlot to faste regler: hver sikkerhetskontroll får en sjekk som beviser at den slår til, og en brannmurregel bekreftes ut fra plasseringen sin og ikke bare ut fra at den finnes.

Selskapets egen git-forge bruker SSH som transport heller enn som en vei inn til et skall. Den ble valgt fordi den styres over SSH og ikke har noe administrasjonspanel på web som må forsvares. Anonym tilgang og tilgang uten nøkkel avvises begge, og hvert repo er privat. SSH-lytteren tar bare imot tilkoblinger på loopback, på port 23231, og folk utenfra når den ved å hoppe via en jump host, så brannmuren åpner ingen ekstra port for den.

Hvor det går galt

En herdingslinje som står i filen mens serveren overser den. For de fleste nøkkelord beholder `sshd` den første verdien den leser, og en `Include` folder ut filene sine i leksikalsk rekkefølge. Når hovedfilen trekker inn en mappe med drop-in-filer først, slår en leverandørfil som sorteres tidlig, din, og begge filene ser

fortsatt riktige ut. Et søk i konfigurasjonen beviser bare at en linje ble skrevet. Spør heller daemonen: `sshd -t` sjekker syntaksen, og `sshd -T` skriver ut innstillingene den faktisk kommer til å bruke.

```
/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 endret vertsnøkkel som blir slettet og godtatt. Når en kjent server viser fram en annen vertsnøkkel, skriver `ssh` en advarsel og nekter å koble til. Refleksen er `ssh-keygen -R` på vertsnavnet, en ny tilkobling og et skrevet «yes», og det er alt en mann i midten trenger. Bekreft det nye fingeravtrykket gjennom en annen kanal først, f.eks. med `ssh-keygen -lf` på serverens offentlige vertsnøkkel fra konsollen. I skript er `StrictHostKeyChecking no` verre enn det ser ut: i tillegg til å lagre nye nøkler i det stille lar det tilkoblinger til verter der nøkkelen er endret, gå videre, innenfor visse grenser. Innstillingen `accept-new` lagrer ukjente verter og nekter fortsatt endrede nøkler.

Innlogging med passord som står åpen mot hele internett. `PasswordAuthentication` står som standard på yes, og det samme gjør tastaturinteraktiv autentisering, som kan be om det samme passordet en annen vei. Når hver bruker har en nøkkel, setter du begge til no. `PermitRootLogin` står allerede som standard på `prohibit-password`, som avviser passord for root, men fortsatt slipper inn en nøkkel for root. Å flytte daemonen bort fra port 22 endrer hvor den lytter, og lar uendret hvem som kan gjette mot den, og hver kontroll som navngir porten, må flyttes med, slik utestengingsjailen over viser.

Videresending av agenten som låner bort nøklene dine til serveren. Med `ForwardAgent` slått på, eller `ssh -A`, kan alle som kan gå forbi filrettighetene på den eksterne verten, root inkludert, bruke agenten din til å autentisere seg andre steder som deg så lenge du er tilkoblet. De kan ikke kopiere nøkkelen, men de kan signere med den. For å nå en maskin bak en bastionvert bruker du i stedet en jump host: `ssh -J bastion.example.com app.internal` videresender en TCP-tilkobling gjennom bastionen og forhandler økten med målet fra ende til ende, så bastionen formidler byte den ikke kan lese, og nøkkelen blir værende på den bærbare maskinen din.

Les mer

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — de tre lagene, de to modellene for tillit til vertsnøkler og sikkerhetsbetraktningene.
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — nøkkelutveksling, verifisering av vertsnøkkelen og versjonsstrengen hver side sender først.
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — hvordan innlogging med offentlig nøkkel, passord og vertsbasert tillit virker på ledningen.
- sshd_config(5) — hvert nøkkelord på serversiden med standardverdien, og regelen om at den første verdien som hentes, er den som brukes.
- ssh_config(5) — klientsiden, bl.a. StrictHostKeyChecking, ProxyJump og ForwardAgent.
- sshd(8) — testmodusene `-t` og `-T` og formatene for authorized_keys og known_hosts.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er SSH?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/ssh/