Engineer

Privilegium

🎟 en rettighet et system gir en person eller en prosess

Hva det er

Et privilegium er lov til å gjøre noe et system ellers forbyr, gitt til en person eller et kjørende program fordi en ansvarlig instans har bestemt at det skal ha det: å lese en annen brukers filer, binde en nettverksport under 1024, opprette kontoer eller kjøre en databasefunksjon. Folk snakker som regel om privilegier i flertall, fordi et subjekt sjelden har bare ett, og om «privilegerte» og «uprivilegerte» kontoer eller prosesser for å si hvilken side av grensen noe står på. NIST-ordlisten legger til den delen som er lett å glemme: privilegier beskriver hva et subjekt har lov til å gjøre, og de defineres av en ansvarlig instans og skrives ned i en policy eller i regler.

Ordet står ved siden av to andre. Tilgangskontroll er selve avgjørelsen for hver forespørsel; et privilegium er en av opplysningene den avgjørelsen slår opp om den som spør. En tilgang er som regel den samme tanken sett fra ressursens side: filen sier hvem som får lese den, og den som leser, har retten til å gjøre det.

På Linux har ordet en presis betydning, fordi det er en systemkjerne som håndhever det for hver eneste prosess. Manualsiden capabilities(7) beskriver den tradisjonelle delingen i Unix. Privilegerte prosesser, de med effektiv bruker-ID 0, kjent som root, går forbi alle tilgangssjekkene i kjernen, mens alle andre prosesser sjekkes mot sine identitetsopplysninger. Siden Linux 2.2 har kjernen delt roots makt opp i separate enheter kalt capabilities, som kan slås av og på hver for seg. Et program som bare trenger å binde en lav port, kan dermed få `CAP_NET_BIND_SERVICE` og ingenting annet. Manualsiden credentials(7) forklarer hvem sin identitet som blir sjekket: den reelle bruker-ID-en sier hvem som eier prosessen, den effektive bruker-ID-en er den kjernen sammenligner med tilgangene, og den lagrede set-user-ID-en lar et program legge bort en hevet identitet og ta den opp igjen senere.

Databaser og organisasjoner bruker samme ord om sine egne tildelinger. En databaserolle har privilegier på skjemaer, tabeller og funksjoner. En administratorkonto har privilegier en vanlig ansattkonto ikke har. Tanken er den samme på hvert nivå: en rett, gitt med vilje, til å krysse en grense som ellers er stengt.

Bruk begrepet i en beslutning

Bruk «Hva er et privilegium?» 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

Privilegiene et program har, er størrelsen på skaden det kan gjøre når noe går galt med det. Det er det uttalte målet med capabilities: et kompromittert program gjør mindre skade med et smalt sett rettigheter enn det samme programmet gjør når det kjører som root. En webserver med en feil som lar en angriper kjøre kode, gir hele maskinen til den som fant feilen hvis den kjører med root-privilegier. Den samme feilen i en prosess som bare kan lese sin egen katalog og binde én port, når svært lite.

NIST kaller dette minste privilegium. Kontroll AC-6 i SP 800-53 ber en organisasjon tillate bare de tilgangene brukere, og prosessene som handler på vegne av dem, trenger for å utføre oppgavene de er tildelt. Utvidelsene til kontrollen gjør det til rutinearbeid: administratorer bruker en uprivilegert konto til hverdagsoppgaver som e-post, privilegerte kontoer begrenses til dem som trenger dem, bruken av privilegerte funksjoner logges, og privilegier gjennomgås og fjernes når ingen lenger kan vise at de trengs.

Standardinnstillingene starter sjelden der. I systemd kjører en systemtjeneste uten en `User=` -linje som root, og `NoNewPrivileges=` , innstillingen som hindrer en tjeneste og barna dens i noen gang å få mer gjennom `execve()` , er av inntil noen slår den på. Databaseverktøy kobler seg ofte til som eier av hver tabell, fordi det var slik tabellene ble opprettet. Nyansatte kopieres ofte fra kontoen til en kollega, med rettigheter og alt. Hver av disse standardene er praktisk den første dagen og dyr den dagen noen bryter seg inn.

Avveiningen er reell. Å stramme inn privilegier tar tid, fordi noen må finne ut hva et program faktisk gjør, og et sett som er for smalt, knekker ting ved kjøretid i stedet for ved installasjon. Beslutningen et team står overfor, er som regel hvor tiden skal brukes først: på alt som vender mot internett, på alt som holder data folk ville savnet, og på kontoene som kan endre de andre kontoene. For en kjøper eller en revisor er det nyttige spørsmålet enkelt og direkte: hvem og hva har administratorrettigheter her, og hvorfor?

Slik ser det ut i praksis

Tretten tjenester på selskapets egen vert kjørte med de privilegiene tjenestestyreren gir enhetsfilene som standard, og det er de fleste. Hver enhetsmal fikk et privilegietak låst til enhetens egne privilegier, et filter for systemkall, begrensede adressefamilier og minne som kan være skrivbart eller kjørbart, men aldri begge deler. Scoren for herding gikk fra 9,6 til 1,5 for dødmannsknappen, fra 8,3 til 1,5 for git-forgen og fra 9,6 til 2,3 og 2,5 for de to enhetene som tar sikkerhetskopier. Ett direktiv viste seg å være feilstavet i tre maler, og tjenestestyreren hadde logget en ukjent nøkkel og startet enheten likevel. Nå kjøres derfor hver enhet gjennom tjenestestyrerens eget verifiseringsverktøy ved hver konvergering. Siden skriver også ned hva scoren ikke ser: et direktiv som ødelegger en enhet, gjør det ved kjøretid.

På en applikasjonsplattform logger API-et inn i databasen som en rolle etter prinsippet om minste privilegium. Rollen har bare EXECUTE på appfunksjonene, og de kjører som SECURITY DEFINER med parameteriserte spørringer. Over databaserollen ligger brukerens egne privilegier: abonnementsnivå, rolle, organisasjon og avgrensning per skip. Dem vurderer serveren for hver enkelt forespørsel, mens det grensesnittet sperrer, bare regnes som UX.

I én organisasjon ble identiteter og tilganger samlet på tvers av M365, 1Password, Red Hat SSO og OKTA SSO. Minste privilegium ble håndhevet, og onboarding og offboarding ble standardisert, slik at tilganger ble trukket tilbake raskt i stedet for å bli hengende igjen etter at noen hadde gått videre.

Hvor det går galt

En tjeneste kjører som root fordi enheten ikke sier noe. Tjenestestyrerens standard for en systemtjeneste er root, så en enhet uten `User=` -linje har alle privilegiene maskinen har. Gi hver tjeneste sin egen bruker, eller la systemd lage

en med `DynamicUser=`, som oppretter brukeren når enheten starter og frigjør den når enheten stopper.

Én enkelt capability viser seg å være nesten hele root. `CAP_SYS_ADMIN` dekker så mange operasjoner at dens egen manualside kaller den overbelastet og «the new root». Et team som gir den bort for å låse opp én funksjon, har gitt bort det meste av root. Finn akkurat den rettigheten programmet trenger, og hold avgrensningssettet (bounding set) til den listen:

```
[Service]
User=web
NoNewPrivileges=yes
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

Med disse innstillingene kan en tjeneste som kjører som vanlig bruker, binde port 80, og alle andre capabilities er fjernet fra avgrensningssettet.

En prosess klatrer høyere enn der den startet. Et program som kjører som vanlig bruker, kan fortsatt kjøre en set-user-ID-binærfil eller en fil med capabilities knyttet til seg, og kjernen hever privilegiene for den kjøringen. De som forsvarer systemet, kjenner det igjen som en prosess der den effektive bruker-ID-en skiller seg fra den reelle. `NoNewPrivileges=yes` stenger den veien for en tjeneste og for hvert barn den starter.

Rettighetene hoper seg opp når folk bytter jobb. Noen flytter til et annet team og beholder den gamle tilgangen ved siden av den nye, og etter noen år har kontoen summen av hver rolle personen noen gang har hatt. AC-6 svarer med jevnlig gjennomgang, og AC-2 med kontoforvaltning knyttet til overflyttinger og avganger, slik at et jobbskifte også blir et skifte i tilgang.

Applikasjonens egen innlogging kan gjøre alt. Når et API kobler seg til som eier av skjemaene, når én enkelt injeksjonsfeil hver tabell. En innloggingsrolle som bare kan kalle funksjonene applikasjonen bruker, gjør den samme feilen til et mye mindre problem.

Les mer

- Privilege, in the NIST CSRC glossary — definisjonen fra CNSSI 4009, med merknaden om at en ansvarlig instans definerer privilegier.
- capabilities(7) — hvordan Linux deler root opp i separate capabilities, og hvordan tråder bærer dem.
- credentials(7) — reell, effektiv, lagret og filsystemets bruker-ID, og hva hver av dem avgjør.
- systemd.exec(5) — `User=`, `DynamicUser=`, `NoNewPrivileges=` og innstillingene for capabilities.
- NIST SP 800-53 Rev. 5 — kontroll AC-6, minste privilegium, og utvidelsene som gjør den til rutine.

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 et privilegium?» 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/privilege/