Engineer

Driftshemmelighet

tilgangsdata et program trenger, som nøkler eller passord

Hva det er

En driftshemmelighet er en verdi som lar programvare, eller personen som drifter den, handle på et system, og som gir den samme makten til hvem som helst andre som får lese den. Passordet i en databasetilkobling, et API-token, den private halvdelen av et SSH-nøkkelpar og passfrasen som krypterer en sikkerhetskopi, er alle driftshemmeligheter. I daglig tale krymper ordet til hemmeligheter, og arbeidet med å lage, lagre, rotere og destruere dem heter på engelsk secrets management.

To ting skiller dem fra koden rundt dem. Den første er hvem som leser dem. Passordet til en person sitter i hukommelsen og tastes inn ved en ledetekst, mens en driftshemmelighet leses av et program, et skript for produksjonssetting eller en planlagt jobb, og ingen trenger å huske den. OWASP Secrets Management Cheat Sheet gir den derfor en egen livssyklus: den lages med det minste privilegiet den trenger, roteres etter en plan, trekkes tilbake når den kan ha lekket, og får en utløpsdato der plattformen tillater det, med levetider fra minutter til år. Brukernes passord står utenfor den faste rotasjonen. Den andre er grensen mellom en hemmelighet og konfigurasjonen som peker på den. Et vertsnavn eller stien til et sikkerhetskopirepo sier hvor noe er. Hemmeligheten er den delen som slipper innehaveren inn, og den delen der hver kopi må telles.

Hemmeligheter ligger også i lag. Lageret som holder dem, trenger en hemmelighet selv, og OWASP-veiledningen påpeker at denne primærhemmeligheten, f.eks. root-påloggingsopplysningene hos en skyleverandør, som regel må oppbevares et annet sted.

Bruk begrepet i en beslutning

Bruk «Hva er en driftshemmelighet?» 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

En hemmelighet virker for den som legger den fram. Et token bærer ingen opplysning om hvem som skulle ha det, så hver kopi utvider kretsen av folk og programmer som kan bruke det. Et kodelager er det verste stedet en hemmelighet kan havne: hver klone bærer hele historikken, så en verdi som er committet én gang, reiser til hver bærbar maskin, hver byggeagent og hvert speil som henter repoet, og den blir liggende i historikken etter at linjen som holdt den, er slettet.

Standardene gjør dette til krav. OWASP Application Security Verification Standard, ASVS, krever en løsning for hemmeligheter, f.eks. et nøkkelhvelv, som lager, lagrer, styrer tilgangen til og destruerer hemmelighetene på serversiden, og slår fast at hemmeligheter ikke skal ligge i kildekoden eller i byggeartefakter. Standarden krever minste privilegium for tilgangen til dem, og på det høyeste nivået at de utløper og roteres. NIST SP 800-53 legger til kontrollutvidelsen IA-5(7): ukrypterte statiske autentikatorer skal ikke bygges inn i applikasjoner eller annen statisk lagring. Veiledningen tetter også et smutthull: en verdi som brukes nøyaktig slik den er lagret, regnes som ukryptert, uansett hvilken fil den ligger i.

Avveiningene er ekte. Et sentralt hemmelighetslager er ett system til som må være oppe akkurat når det trengs mest, og OWASP-veiledningen minner om at folk under en driftshendelse forventer nye påloggingsopplysninger raskt. Å dele én hemmelighet mellom flere tjenester sparer arbeid og koster sporbarhet, for etter en lekkasje er det umulig å si hvilken innehaver som mistet den. Manuell håndtering vokser med hver ny hemmelighet, og derfor anbefaler veiledningen automatisk rotasjon. Den advarer også om at en omstart av applikasjonen lar en stjålet påloggingsopplysning fortsette å virke. Først når den trekkes tilbake, slutter den å virke.

Slik ser det ut i praksis

Dette nettstedet skriver regelen sin på siden om hvordan selskapet jobber sikkert: hemmeligheter skal holdes utenfor repoet. Påloggingsopplysninger ligger i et hemmelighetslager eller i laget som håndterer produksjonssetting, utenfor kildekoden og utenfor alt et bygg produserer, og en automatisk skanning kjører ved hver commit og stopper commiten når den finner noe.

Utviklernotatene for repoet forklarer hvordan. Skanneren er gitleaks, som går over arbeidstreet ved hver commit og en gang til i den fulle kvalitetskontrollen, og et treff er en hard stopp som ingen skal fjerne med en unntaksliste. Avhengighetsmappen og de genererte utdatatrærne er unntatt, fordi de er kopier som bygges på nytt fra bunnen av. Notatene beskriver også en stillere feil: når en fil forsvinner midt i skanningen, skriver gitleaks ut en feillinje og avslutter likevel som vellykket, så oppgaven som kjører den, feiler på enhver feillinje. Selve nettstedet er statisk og holder ingen applikasjonshemmeligheter, og det produksjonssettes over SSH med nøkler eieren selv har.

Når hemmelighetene holdes utenfor repoet, havner de et sted repoets egne sjekker ikke når, og et preflight-play for hemmeligheter tar seg av det gapet. Hemmelighetene var nettopp blitt delt opp per vert, og koden var riktig, men operatørens krypterte vault, som ligger på operatørens egen maskin og utenfor synsfeltet til kvalitetsporten, holdt fortsatt den gamle verdien fra da det fantes én vert. Den første kjøringen nådde verten i drift, kjørte åtte oppgaver og nektet på den niende, etter at den allerede hadde skrevet swap-innstillinger til produksjon. Svaret ble et play som ikke kontakter noen vert og ikke samler facts, men kjører de samme to oppgavefilene som den virkelige kjøringen mot vaulten, på omtrent et sekund, før provisjoneringen gjør noe annet. Å gjenbruke koden i stedet for å skrive regelen på nytt hindrer at sjekken og kjøringen glir fra hverandre. En dårlig hemmelighet koster nå en nektelse på to sekunder på en bærbar maskin.

Selve oppdelingen er overgangen til én verdi per vert for hver driftshemmelighet. Plattformen hadde én vert og skulle snart få to, og flere hemmeligheter var skrevet som enkeltverdier, bl.a. sikkerhetskopirepoet, passfrasen til det og en varslingsknapp. Hver av dem ble en oppslagstabell med vertsnavnet i inventaret som nøkkel, og to begrunnelser står ved siden av konfigurasjonen. En dødmannsknapp som deles av to verter, holder seg grønn så lenge én av dem fortsatt melder seg, så den andre verten skjuler at den første er død. Separate restic-repoer kan ha forskjellige hovednøkler selv om de bruker samme passfrase. Men den som får tak i passfrasen og kan lese de krypterte

dataene og nøkkelfilene i begge repoene, kan låse opp begge. Å dele passfrasen knytter dermed eksponeringen deres sammen i stedet for å isolere vertene. Den eksisterende verten står i alle fire tabellene med sikkerhetskopirepoet sitt og begge knappene tomme. Det sier at koblingen finnes, og at verdien fortsatt er operatørens halvdel av arbeidet.

Hvor det går galt

En hemmelighet i en commit. En skanner flagger en verdi i en diff, eller noen oppdager et token i en fil som er pushet, fordi det lå i en konfigurasjonsfil eller testdata da commiten ble laget. OWASP-veiledningen gir rekkefølgen for reparasjonen: trekk tilbake den eksponerte verdien først, roter den deretter, og fjern den først da fra koden eller loggene. Å skrive om historikken for å fjerne commiten endrer alle senere commits og bryter lenker til dem, så det krever planlegging. Oppdagelsen hører hjemme før commiten, i en pre-commit-hook eller i editoren, der et treff ikke koster noe.

En skanning som går forbi filer den aldri leste. En kjøring ender grønn mens en del av treet forble uundersøkt: en fil forsvant underveis, et unntak ble trukket for bredt, eller hemmeligheten har ingen gjenkjennelig form. Mønsterregler fanger tokens med fast format og overser et passord en person har funnet på, og derfor foreslår veiledningen mer enn ett verktøy. Behandle hver feil fra skanneren som et brudd, og hold unntakene til kopier som et bygg lager på nytt.

Én hemmelighet, mange innehavere. To verter, to tjenester eller et helt team bruker den samme verdien. Når den lekker, kan loggene ikke si hvilken kopi som forsvant, og å bytte den betyr å røre alle innehaverne på en gang. SP 800-53 krever at autentikatorene til gruppe- og rollekontoer byttes hver gang medlemskapet endres, og én hemmelighet per innehaver gjør det til ett bytte i stedet for et koordinert.

Hemmeligheten som følger med bygget eller havner i en logg. Et containerbilde, et bundlet skript eller en feilsøkingslinje bærer verdien ut av stedet der den skulle bli. ASVS nevner byggeartefakter ved siden av kildekoden, og OWASP-veiledningen ber om at hemmeligheter holdes utenfor logger og maskeres der en linje må vise til en. Skann det bygget produserer, i tillegg til repoet.

Les mer

- OWASP Secrets Management Cheat Sheet — livssyklusen til en hemmelighet, oppdagelse før commiten og rekkefølgen for reparasjon etter en lekkasje.
- OWASP Application Security Verification Standard — kravene til håndtering av hemmeligheter i gruppe 13.3: et hvelv, minste privilegium og rotasjon.
- NIST SP 800-53 Rev. 5 — håndtering av autentikatorer i IA-5, med IA-5(7) mot innebygde statiske autentikatorer.
- Gitleaks — skanneren dette nettstedet kjører over arbeidstreet sitt ved hver commit.

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 en driftshemmelighet?» 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/secret/