Engineer

Driftshemmelighed

en nøgle eller adgangskode, et program skal bruge

Hvad det er

En driftshemmelighed er en værdi, der lader software, eller den person, der driver den, handle på et system, og som giver enhver anden, der læser den, den samme magt. Adgangskoden til en databaseforbindelse, et API-token, den private halvdel af et SSH-nøglepar og den adgangsfrase, der krypterer en backup, er alle driftshemmeligheder. Til daglig siger man bare hemmeligheder, eller secrets på engelsk, og arbejdet med at oprette, opbevare, rotere og destruere dem kaldes secrets management.

To ting adskiller dem fra koden omkring dem. Den første er læseren. En brugers adgangskode sidder i et menneskes hukommelse og tastes ved en prompt, mens en driftshemmelighed læses af et program, et udrulningsscript eller et planlagt job, og ingen behøver at huske den. OWASP's Secrets Management Cheat Sheet giver den derfor sin egen livscyklus: oprettet med de mindste rettigheder, den har brug for, roteret efter en plan, tilbagekaldt, når den kan være lækket, og sat til at udløbe, hvor platformen tillader det, med levetider fra minutter til år. Brugeres adgangskoder står uden for den faste rotation. Den anden er grænsen mellem en hemmelighed og den konfiguration, der peger på den. Et værtsnavn eller stien til et backuplager siger, hvor noget er. Hemmeligheden er den del, der lukker indehaveren ind, og den del, hvis kopier skal tælles.

Hemmeligheder ligger også i lag. Det lager, der rummer dem, har brug for sin egen hemmelighed, og cheat sheetet bemærker, at den primære hemmelighed, for eksempel en cloududbyders root-adgangsoplysninger, som regel må opbevares et andet sted.

Brug begrebet i en beslutning

Brug »Hvad er en driftshemmelighed?« 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

En hemmelighed virker for den, der viser den frem. Et token bærer ingen oplysning om, hvem der skulle have det, så hver kopi udvider kredsen af mennesker og programmer, der kan bruge det. Et repository med kildekode er det værste sted, en kan havne: hver klon bærer hele historikken, så en værdi, der er committet én gang, rejser ud til hver laptop, hver build-runner og hvert spejl, der henter, og den bliver i historikken, efter at linjen med den er slettet.

Standarderne gør det til krav. OWASP Application Security Verification Standard beder om en løsning til hemmelighedshåndtering, for eksempel en key vault, der opretter, opbevarer, styrer adgangen til og destruerer backend-hemmeligheder, og fastslår, at hemmeligheder ikke må indgå i applikationens kildekode eller i byggeartefakter. Den beder om mindste rettigheder for adgangen til dem og på sit højeste niveau om, at de udløber og roteres. NIST SP 800-53 tilføjer kontroludvidelsen IA-5(7): ukrypterede statiske autentifikatorer må ikke indlejres i applikationer eller andet statisk lager. Vejledningen lukker også et smuthul, for en værdi, der bruges præcis, som den er gemt, regnes for ukrypteret, uanset hvilken fil den ligger i.

Afvejningerne er reelle. Et centralt lager er endnu et system, der skal være oppe, når der er mest brug for det, og cheat sheetet påpeger, at folk under en hændelse forventer hurtigt at få nye adgangsoplysninger. At dele én hemmelighed mellem flere tjenester sparer arbejde og koster sporbarhed, for efter et læk er der ingen måde at se, hvilken indehaver der mistede den. Manuel håndtering vokser med hver hemmelighed, og derfor foretrækker cheat sheetet automatisk rotation og advarer om, at en genstart af programmet lader en stjålet adgangsoplysning virke videre. Kun en tilbagekaldelse stopper den.

Sådan ser det ud i praksis

Sitet skriver sin regel på siden om, hvordan virksomheden arbejder sikkert: hemmeligheder holdes ude af kodebasen. Adgangsoplysninger ligger i en hemmelighedstjeneste eller i udrulningslaget, uden for kildekoden og uden for alt, et build producerer, og en automatisk scanning kører ved hvert commit og standser committet, når den gør et fund.

Udviklingsnoterne for kodebasen udfylder resten. Scanneren er gitleaks, som kører over arbejdstræet ved hvert commit og igen i det fulde kvalitetstjek, og et fund er et hårdt stop, som ingen skal snige sig forbi med en undtagelse. Mappen med afhængigheder og de genererede output-træer er undtaget, fordi de er kopier, der bygges forfra. Noterne beskriver også en stille fejl: når en fil forsvinder midt i scanningen, skriver gitleaks en fejllinje og afslutter alligevel med succes, så den opgave, der kører scanneren, fejler på enhver fejllinje. Selve sitet er statisk og har ingen applikationshemmeligheder, og det udrulles over SSH med nøgler, som ejeren har.

Når hemmeligheder holdes ude af kodebasen, flytter de et sted hen, som kodebasens tjek ikke når, og et preflight-play for hemmeligheder tager sig af det hul. Driftshemmelighederne var lige blevet delt op pr. vært, og koden var korrekt, men operatørens krypterede vault, der bor på operatørens egen maskine og uden for kvalitetsgatens synsfelt, rummede stadig den gamle værdi fra tiden med én vært. Det første converge nåede den kørende vært, kørte otte tasks og afviste på den niende, efter at det allerede havde skrevet swap-indstillinger til produktion. Svaret blev et play, der hverken kontakter en vært eller indsamler facts, men kører de samme to task-filer som det rigtige converge mod vaulten, på omkring et sekund og før provisioneringen gør noget andet. Når koden genbruges i stedet for at reglen gentages, holdes tjekket og converge-kørslen på linje. En forkert hemmelighed koster nu en afvisning på to sekunder på en laptop.

Selve opdelingen er skiftet til én værdi pr. vært for hver driftshemmelighed. Platformen havde én vært og skulle snart have to, og flere hemmeligheder var skrevet som enkeltværdier, blandt dem backup-repositoriet, dets adgangsfrase og en alarmswitch. Hver blev en tabel med inventarets værtsnavn som nøgle, og to begrundelser står ved siden af konfigurationen. En dead man's switch, som to værter deler, forbliver grøn, så længe den ene stadig melder sig, så den anden vært skjuler, at den første er død. Separate restic-lagre kan have forskellige hovednøgler, selv når de bruger samme adgangsfrase. Men en person, der får fat i den adgangsfrase og kan læse begge lagres krypterede data og nøglefiler, kan

låse begge op. Delingen udsætter dermed begge lagre for risiko ved samme læk i stedet for at isolere værterne. Den eksisterende vært står i alle fire tabeller med sit lager og begge switches uden værdi, og det siger, at koblingen findes, mens operatøren stadig skal levere værdien.

Hvor det går galt

En hemmelighed havner i et commit. En scanner markerer en værdi i en diff, eller nogen opdager et token i en pushet fil, fordi det lå i en konfigurationsfil eller en testfixture, da committet blev lavet. OWASP's cheat sheet giver rækkefølgen for oprydningen: tilbagekald den lækkede værdi først, roter den så, og fjern den først derefter fra koden eller loggene. At omskrive historikken for at fjerne committet ændrer alle senere commits og bryder links til dem, så det kræver planlægning. Opdagelsen hører hjemme før committet, i en pre-commit-hook eller i editoren, hvor et fund intet koster.

Scanningen går hen over filer, den aldrig læste. En kørsel ender grøn, mens en del af træet aldrig blev undersøgt: en fil forsvandt undervejs, en undtagelse blev for bred, eller hemmeligheden har ingen genkendelig form. Mønsterregler fanger tokens med et fast format og overser en adgangskode, et menneske har fundet på, og derfor foreslår cheat sheetet mere end ét værktøj. Behandl enhver fejl fra scanneren som en fejlet kørsel, og begræns undtagelserne til kopier, som et build genskaber.

Én hemmelighed har mange indehavere. To værter, to tjenester eller et helt team bruger den samme værdi. Når den lækker, kan loggene ikke sige, hvilken kopi der slap ud, og en udskiftning kræver, at alle indehavere røres på én gang. SP 800-53 beder om, at autentifikatorer til gruppe- og rollekonti skiftes, hver gang medlemskabet ændrer sig, og med én hemmelighed pr. indehaver bliver det én udskiftning i stedet for en koordineret.

Hemmeligheden følger med buildet eller havner i en log. Et container-image, et bundtet script eller en debuglinje bærer værdien ud af det sted, den skulle blive. ASVS nævner byggeartefakter ved siden af kildekode, og cheat sheetet beder om, at hemmeligheder holdes ude af logge og maskeres, hvor en linje er nødt til at henvise til en. Scan det, buildet producerer, og ikke kun kodebasen.

Læs mere

- OWASP Secrets Management Cheat Sheet — en hemmeligheds livscyklus, opdagelse før committet og rækkefølgen for oprydning efter et læk.
- OWASP Application Security Verification Standard — kravene til hemmelighedshåndtering i gruppe 13.3: en vault, mindste rettigheder og rotation.
- NIST SP 800-53 Rev. 5 — håndtering af autentifikatorer i IA-5, med IA-5(7) mod indlejrede statiske autentifikatorer.
- Gitleaks — den scanner, sitet kører over sit arbejdstræ ved hvert commit.

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 en driftshemmelighed?« 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/secret/