Engineer

Експлуатаційний секрет

ключ чи пароль, потрібний програмі для роботи

Що це таке

Експлуатаційний секрет — це значення, яке дає програмі або людині, що її обслуговує, змогу діяти в системі і дає ту саму силу кожному, хто його прочитає. Пароль до з'єднання з базою даних, API-токен, приватна половина пари SSH-ключів і парольна фраза, якою зашифровано резервну копію, — усе це експлуатаційні секрети. У повсякденній мові кажуть просто секрети, а роботу зі створення, зберігання, ротації та знищення секретів називають керуванням секретами (secrets management).

Від коду довкола них їх відрізняють дві речі. Перша — хто їх читає. Пароль користувача зберігається в пам'яті людини, і його вводять у відповідь на запит, а експлуатаційний секрет читає програма, скрипт розгортання чи завдання за розкладом, і пам'ятати його нікому не треба. Тому OWASP Secrets Management Cheat Sheet описує для нього окремий життєвий цикл: його створюють із мінімальними привілеями, яких він потребує, змінюють за графіком, відкликають, коли він міг витекти, і обмежують строком дії там, де платформа це дозволяє, а тривалість життя коливається від хвилин до років. Паролі користувачів у цю регулярну ротацію не входять. Друга — межа між секретом і конфігурацією, яка на нього вказує. Ім'я хоста чи шлях до репозиторію резервних копій кажуть, де щось лежить. Секрет — це та частина, яка впускає власника, і та, копії якої треба рахувати.

Секрети ще й вкладені один в одного. Сховищу, де вони лежать, потрібен власний секрет, і шпаргалка OWASP зазначає, що цей первинний секрет, наприклад кореневі облікові дані хмарного провайдера, зазвичай доводиться тримати деінде.

Застосуйте поняття до рішення

Застосуйте «Що таке експлуатаційний секрет?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Секрет працює на того, хто його пред’являє. Токен не зберігає відомостей про те, кому його призначали, тож кожна копія розширює коло людей і програм, які можуть ним скористатися. Репозиторій вихідного коду — найгірше місце, куди він може потрапити: кожен клон несе всю історію, тож значення, закомічене один раз, розходиться на кожен ноутбук, раннер збірки й дзеркало, що його завантажує, і лишається в історії після того, як рядок із ним видалено.

Стандарти перетворюють це на вимоги. OWASP Application Security Verification Standard вимагає рішення для керування секретами, наприклад сховища ключів (key vault), яке створює й зберігає серверні секрети, контролює доступ до них і знищує їх, і прямо пише, що секрети не можна включати у вихідний код застосунку чи в артефакти збірки. Він вимагає мінімальних привілеїв для доступу до них, а на найвищому рівні — щоб строк їхньої дії спливав і їх змінювали. NIST SP 800-53 додає посилення контролю IA-5(7): незашифровані статичні автентифікатори не можна вбудовувати в застосунки чи інше статичне сховище. Настанова закриває й лазівку: значення, яке використовують точно в тому вигляді, у якому його збережено, вважається незашифрованим, хай у якому файлі воно лежить.

Компроміси реальні. Центральне сховище — ще одна система, яка має працювати саме тоді, коли вона найпотрібніша, і шпаргалка нагадує, що під час інциденту люди чекають нових облікових даних швидко. Спільний секрет для кількох сервісів економить роботу, але коштує простежуваності, бо після витоку неможливо сказати, хто з власників його втратив. Ручна робота зростає з кожним секретом, тому шпаргалка радить автоматичну ротацію і попереджає, що перезапуск застосунку лишає викрадені облікові дані чинними. Зупиняє їх лише відкликання.

Як це виглядає на практиці

Сайт формулює своє правило на сторінці про те, як компанія працює безпечно: секрети тримають поза репозиторієм. Облікові дані лежать у сховищі секретів або на рівні розгортання, поза вихідним кодом і поза всім, що створює збірка, а автоматичне сканування перевіряє кожен коміт і зупиняє його, щойно щось знаходить.

Нотатки для розробників цього репозиторію додають подробиці. Сканер — gitleaks, він проходить робоче дерево під час кожного коміту й ще раз у повній перевірці якості, і знахідка означає жорстку зупинку, яку не обходять винятком. Каталог залежностей і згенеровані дерева виводу виключено зі сканування, бо це копії, які збірка відтворює з нуля. Нотатки фіксують і тихіший збій: коли файл зникає посеред сканування, gitleaks друкує рядок із помилкою і все одно завершується успішно, тому завдання, яке запускає сканер, падає на будь-якому такому рядку. Сам сайт статичний і секретів застосунку не має, а розгортають його через SSH ключами, які тримає власник.

Коли секрети тримають поза репозиторієм, вони опиняються там, куди перевірки репозиторію не дістають, і попередній плей для секретів закриває цю прогалину. Секрети щойно розділили за хостами, і код був правильний, але зашифроване сховище секретів оператора, що навмисно живе лише на його машині й поза полем зору перевірки якості, досі тримало старе значення, розраховане на один хост. Перший прогін converge дістався живого хоста, виконав вісім завдань і відмовився на дев'ятому, коли вже записав на продакшн налаштування свопу. Відповіддю став плей, що не звертається до жодного хоста, не збирає фактів і проганяє на сховищі ті самі два файли завдань, що й справжній converge, приблизно за секунду і раніше за будь-який інший крок підготовки хоста. Оскільки код той самий, а не переказ правила, перевірка й converge не розходяться. Хибний секрет тепер коштує двосекундної відмови на ноутбуці.

Сам поділ описано на сторінці про те, як кожен експлуатаційний секрет отримав окреме значення для кожного хоста. Платформа мала один хост і от-от мала отримати другий, а кілька секретів мали лише одне значення на всю платформу, серед них репозиторій резервних копій, його парольна фраза й перемикач тривоги. Кожне стало словником із ключем за ім'ям хоста в інвентарі, і поруч із конфігурацією записано дві причини. Dead man's switch, спільний для двох хостів, лишається зеленим, поки бодай один із них подає сигнал, тож другий хост приховує смерть першого. Окремі репозиторії

restic можуть мати різні головні ключі, навіть якщо використовують ту саму парольну фразу. Проте той, хто отримає цю парольну фразу й зможе прочитати зашифровані дані та файли ключів обох репозиторіїв, зможе відкрити обидва. Отже, спільна парольна фраза пов'язує ризики розкриття даних обох репозиторіїв замість того, щоб ізолювати хости. Наявний хост внесено в усі чотири словники, залишивши порожніми його репозиторій та обидва перемикачі: обв'язка вже є, а значення ще має надати оператор.

Де виникають проблеми

Секрет потрапляє в коміт. Сканер позначає значення в diff, або хтось помічає токен у запушеному файлі, бо той лежав у файлі конфігурації чи тестовій фікстурі, коли робили коміт. Шпаргалка OWASP задає порядок дій: спершу відкликати скомпрометоване значення, потім змінити його, і лише тоді видалити з коду чи журналів. Переписування історії, щоб прибрати коміт, змінює всі пізніші коміти й ламає посилання на них, тож його треба планувати. Виявляти такі речі слід до коміту, у pre-commit-хуку чи в редакторі, де знахідка нічого не коштує.

Сканування проходить повз файли, яких не прочитало. Прогін завершується зеленим, хоча частину дерева так і не перевірено: файл зник посеред обходу, виняток вийшов заширокий або секрет не має впізнаваної форми. Правила-шаблони добре ловлять токени фіксованого формату й пропускають пароль, який вигадала людина, тому шпаргалка радить більше ніж один інструмент. Вважайте будь-яку помилку сканера провалом прогону і лишайте у винятках тільки копії, які відтворює збірка.

Один секрет на багатьох власників. Два хости, два сервіси чи ціла команда користуються тим самим значенням. Коли воно витікає, журнали не можуть сказати, яка копія пішла, а заміна вимагає торкнутися всіх власників одночасно. SP 800-53 вимагає змінювати автентифікатори групових і рольових облікових записів щоразу, коли змінюється їхній склад, а окремий секрет для кожного власника робить це однією заміною замість узгодженої.

Секрет їде разом зі збіркою або потрапляє в журнал. Образ контейнера, зібраний скрипт чи налагоджувальний рядок виносить значення за межі місця, де воно мало лишатися. ASVS згадує артефакти збірки поруч із вихідним кодом, а шпаргалка просить тримати секрети поза журналами й маскувати їх там, де рядок мусить на них посилатися. Скануйте те, що створює збірка, а не лише репозиторій.

Що почитати далі

- OWASP Secrets Management Cheat Sheet — життєвий цикл секрету, виявлення до коміту й порядок дій після витоку.
- OWASP Application Security Verification Standard — вимоги до керування секретами в групі 13.3: сховище, мінімальні привілеї й ротація.
- NIST SP 800-53 Rev. 5 — керування автентифікаторами в IA-5 і IA-5(7) проти вбудованих статичних автентифікаторів.
- Gitleaks — сканер, який сайт запускає по своєму робочому дереву під час кожного коміту.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке експлуатаційний секрет?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/secret/