Engineer

Привілей

🎟 право, яке система надає людині чи процесу

Що це таке

Привілей — це дозвіл робити те, що система інакше забороняє. Його має людина або запущена програма, бо так вирішив той, хто має на це повноваження: читати чужі файли, слухати порт нижче 1024, створювати облікові записи чи викликати функцію в базі даних. Зазвичай говорять про набір привілеїв, бо суб'єкт рідко має лише один, а також про «привілейовані» та «непривілейовані» облікові записи й процеси, щоб показати, з якого боку межі вони стоять. Із привілеями працюють на кожному рівні, і глосарій NIST додає те, що легко забути: привілеї описують дозволену поведінку суб'єкта, а визначає їх уповноважена сторона й записує в політиці чи правилах.

Поруч із цим словом стоять ще два. Контроль доступу — це саме рішення щодо кожного запиту, а привілей — один із фактів про того, хто питає, на які це рішення спирається. Дозвіл (permission) зазвичай означає те саме, але з боку ресурсу: файл каже, хто може його читати, а читач має на це право.

У Linux слово має точне значення. Сторінка посібника capabilities(7) описує класичний поділ Unix: привілейовані процеси, ефективний ідентифікатор користувача яких дорівнює 0 і які називають root, обходять усі перевірки дозволів у ядрі, а кожен інший процес перевіряється за його обліковими даними. Починаючи з Linux 2.2, ядро ділить повноваження root на окремі одиниці, capabilities, які можна вмикати й вимикати по одній. Тож програмі, якій треба лише слухати низький порт, можна дати `CAP_NET_BIND_SERVICE` і нічого більше. Сторінка credentials(7) пояснює, чию саме ідентичність перевіряють: реальний ідентифікатор показує, хто володіє процесом, ефективний — це те, що ядро порівнює з дозволами, а збережений set-user-ID дає програмі змогу відкласти підвищену ідентичність і пізніше повернутися до неї.

Бази даних і організації вживають те саме слово для власних надань. Роль у базі даних має привілеї на схеми, таблиці й функції. Обліковий запис адміністратора має привілеї, яких немає у звичайного працівника. Ідея на кожному рівні та сама: право, надане свідомо, перетнути межу, яка інакше закрита.

Застосуйте поняття до рішення

Застосуйте «Що таке привілей?» 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

Привілеї програми — це розмір шкоди, яку вона може завдати, коли з нею щось трапиться. Саме таку мету мають capabilities: скомпрометована програма з вузьким набором прав шкодить менше, ніж та сама програма, що працює як root. Вебсервер із вразливістю, яка дає виконати код, і з правами root віддає всю машину тому, хто знайшов вразливість. Та сама вразливість у процесі, що може читати лише свій каталог і слухати один порт, дістає дуже мало.

NIST називає це принципом найменших привілеїв. Контроль AC-6 у SP 800-53 вимагає, щоб організація дозволяла лише ті доступи, які потрібні користувачам і процесам, що діють від їхнього імені, для виконання їхніх завдань. Розширення цього контролю перетворюють принцип на щоденну роботу: адміністратори виконують звичайні справи, як-от пошту, під непривілейованим обліковим записом, привілейовані записи мають лише ті, кому вони потрібні, використання привілейованих функцій журналюється, а привілеї переглядають і забирають, коли потребу в них більше неможливо обґрунтувати.

Типові налаштування рідко починаються з цього. У systemd системний сервіс без рядка `User=` працює як root, а `NoNewPrivileges=`, параметр, який забороняє сервісу та його дочірнім процесам отримувати більше прав через `execve()`, вимкнений, доки хтось його не ввімкне. Інструменти для баз даних часто під’єднуються як власник усіх таблиць, бо саме так створювали

схему. Нових працівників часто копіюють з облікового запису колеги разом з усіма правами. Кожне таке налаштування зручне першого дня і дороге того дня, коли стається злам.

Компроміс тут справжній. Звуження привілеїв потребує часу, бо хтось має з'ясувати, що програма насправді робить, а надто вузький набір ламає роботу під час виконання, а не під час встановлення. Тож команда зазвичай вирішує, куди спершу вкласти цей час: у все, що дивиться в інтернет, у все, що зберігає дані, втрата яких буде відчутною, і в облікові записи, які можуть змінювати інші облікові записи. Для покупця чи аудитора корисне запитання просте: хто і що має тут права адміністратора і чому?

Як це виглядає на практиці

На власному сервері компанії тринадцять unit-файлів працювали з тими привілеями, які наглядач дає за замовчуванням, а це більшість із них. Кожен шаблон отримав стелю привілеїв на рівні самого юніта, фільтр системних викликів, обмежені родини адрес і пам'ять, яка може бути доступною для запису або для виконання, але ніколи для обох одночасно. Оцінка захищеності перемикача мертвої руки знизилася з 9,6 до 1,5, git-форджа — з 8,3 до 1,5, а двох юнітів резервного копіювання — з 9,6 до 2,3 і 2,5. В одній із директив у трьох шаблонах виявилася помилка в написанні, і наглядач записував у журнал невідомий ключ та все одно запускав юніт. Тому тепер під час кожного застосування конфігурації кожен юніт проходить власну перевірку наглядача. Сторінка також фіксує, чого оцінка не враховує: директива, яка ламає юніт, робить це під час роботи.

На платформі застосунку API входить у базу даних під роллю з найменшими привілеями, яка може лише виконувати EXECUTE для функцій застосунку, а ті працюють як SECURITY DEFINER із параметризованими запитами. Права на кшталт тарифу, ролі, організації та прив'язки до судна перевіряються на сервері при кожному запиті, а обмеження в інтерфейсі вважаються лише частиною UX.

В одній організації ідентичності та доступи зібрали разом у M365, 1Password, Red Hat SSO і OKTA SSO, застосували принцип найменших привілеїв і стандартизували прийом та звільнення працівників, тож доступ відкликали одразу, а не залишали після того, як людина переходила на іншу роботу.

Де виникають проблеми

Сервіс працює як root, бо його юніт про користувача мовчить. Для системного сервісу наглядач за замовчуванням обирає root, тож юніт без рядка `User=` має всі привілеї машини. Дайте кожному сервісу власного користувача або дозвольте systemd виділити його через `DynamicUser=`: користувач з'являється, коли юніт стартує, і зникає, коли юніт зупиняється.

Одна capability виявляється майже цілим root. `CAP_SYS_ADMIN` охоплює так багато операцій, що її власний посібник називає її перевантаженою і «the new root». Команда, яка надає її, щоб відкрити одну функцію, фактично надає більшу частину root. Знайдіть конкретну capability, яка потрібна програмі, і обмежте bounding set саме нею:

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

Ці налаштування дають сервісу, що працює від звичайного користувача, слухати порт 80, а всі інші capabilities прибирають із його bounding set.

Процес піднімається вище, ніж стартував. Програма від звичайного користувача все ще може запустити файл із set-user-ID або файл із прикріпленими capabilities, і тоді ядро підвищує її привілеї на час цього запуску. Захисники розпізнають це як процес, у якого ефективний ідентифікатор користувача відрізняється від реального. `NoNewPrivileges=yes` закриває цей шлях для сервісу і кожного процесу, який він запускає.

Права накопичуються, коли люди змінюють посади. Людина переходить в іншу команду й зберігає старий доступ поруч із новим, а за кілька років її обліковий запис має суму всіх ролей, які вона колись мала. AC-6 відповідає на це періодичним переглядом, а AC-2 — керуванням обліковими записами, прив'язаним до переведень і звільнень, щоб зміна посади означала й зміну доступу.

Власний вхід застосунку може все. Коли API під'єднується як власник схеми, одна вразливість до ін'єкції дістає кожну таблицю. Роль для входу, яка може лише викликати функції, потрібні застосунку, робить ту саму вразливість значно меншою проблемою.

Що почитати далі

- Privilege у глосарії NIST CSRC — визначення з CNSSI 4009 із приміткою, що привілеї визначає уповноважена сторона.
- capabilities(7) — як Linux ділить root на окремі capabilities і як їх несуть потоки.
- credentials(7) — реальний, ефективний, збережений і файловий ідентифікатори користувача та що вирішує кожен із них.
- systemd.exec(5) — `User=`, `DynamicUser=`, `NoNewPrivileges=` і параметри для capabilities.
- NIST SP 800-53 Rev. 5 — контроль AC-6 про найменші привілеї та розширення, що роблять його рутиною.

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 привілей?» 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/privilege/