Engineer

Контроль доступу

рішення, чи дозволити кожен запит до ресурсу

Що це таке

Контроль доступу — це частина системи, що багато разів на секунду відповідає на одне запитання: чи може той, хто звертається, зробити ось це з ось цим ресурсом просто зараз? Звертатися може людина, сервіс або завдання за розкладом. Ресурсом може бути файл, рядок у базі даних, кінцева точка API, git-репозиторій чи двері. Коли йдеться про всю систему правил і засобів контролю доступу в організації, часто кажуть «керування доступом», а глосарій NIST називає синонімом усієї ідеї авторизацію. Його визначення охоплюють обидві половини роботи: рішення дозволити чи заборонити суб'єктові доступ до об'єктів системи і процес, у якому конкретні запити на отримання й використання інформації та пов'язаних із нею сервісів обробки або на вхід до певних фізичних приміщень задовольняють чи відхиляють.

Його легко сплутати з автентифікацією, але це два окремі кроки. NIST SP 800-162 каже це прямо: автентифікація — не те саме, що контроль доступу. Автентифікація встановлює, хто звертається. Контроль доступу бере цю ідентичність разом з усім іншим, що відомо про запит, і вирішує, що цьому суб'єктові дозволено. Система може бездоганно перевірити людину й однаково дозволити їй читати чужі записи, бо другого запитання ніхто так і не поставив.

Той самий посібник ділить роботу на дві частини. Точка ухвалення рішень обчислює відповідь, а точка застосування політики стоїть на шляху запиту й виконує цю відповідь. Вони можуть жити в одному процесі, але розрізняти їх варто, бо рішення, яке ніщо не втілює, нічого не захищає. NIST SP 800-53 закріплює це у двох контролях: AC-24 вимагає, щоб рішення застосовували до кожного запиту перед втіленням, а AC-3 — щоб затверджені авторизації справді втілювалися.

Правила бувають різної форми: список дозволених ідентичностей на кожному об’єкті, ролі, як у RBAC, або атрибути того, хто звертається, об’єкта, дії та середовища, яким присвячено SP 800-162. Хай яка форма, щось однаково має ставити запитання під час кожного запиту.

Застосуйте поняття до рішення

Застосуйте «Що таке контроль доступу?» 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

Щойно людина ввійшла, майже все, до чого вона може дістатися, визначає контроль доступу. Сильна автентифікація перед слабким контролем доступу лише фіксує, який обліковий запис забрав дані. Для компанії, яка зберігає чужу інформацію, саме тут обіцянка про те, хто що бачить, або тримається, або ні.

NIST описує ідеальний механізм втілення як монітор звернень. Контроль AC-25 у SP 800-53 вимагає такого, що захищений від втручання, викликається завжди й досить малий, щоб його можна було проаналізувати й перевірити. Ці три властивості — практичний тест для будь-якої перевірки в справжньому продукті. Якщо той, хто звертається, може змінити дані, на які спирається перевірка, вона не проходить першу умову. Якщо до даних веде шлях в обхід перевірки, вона не проходить другу. Якщо логіку розкидано по сотнях обробників і ніхто не може прочитати її від початку до кінця, вона не проходить третю.

OWASP Application Security Verification Standard перетворює це на вимоги, за якими команда може перевіряти свій продукт. Розділ про авторизацію вимагає обмежити доступ до функцій тими, хто має явний дозвіл, так само обмежити доступ до конкретних елементів даних і застосовувати правила на довіреному рівні сервісу, а не покладатися на те, чим може маніпулювати недовірений клієнт, як-от JavaScript у браузері. Він також вимагає, щоб зміна

значень, на яких ґрунтується рішення, діяла одразу, а там, де це неможливо, щоб компенсаційний засіб піднімав тривогу, коли хтось діє, уже втративши на це право.

Компроміс — це ціна на шляху запиту. Кожне рішення потребує фактів, а кожен факт — звернення до сховища. Коли це звернення пропускають, перевірка зазвичай опиняється там, де вона найдешевша, а часто це місце, яке контролює сам той, хто звертається.

Як це виглядає на практиці

Одна людина на морській платформі може діяти від імені кількох організацій, тому активну організацію зберігають на клієнті, але не довіряють цьому значенню, вирішуючи питання доступу. OrganizationContext зберігає поточний вибір у localStorage, тож перемикання миттєве, а серверний cookie дублює його, щоб сторінки, які рендеряться на сервері, отримували той самий контекст. Авторизацію повторно перевіряють на сервері під час кожного запиту. Сторінка проводить межу одним реченням: вирішувати, що користувачеві дозволено робити, може лише сервер, а хто підмінить вміст localStorage, той змінить тільки те, що показує йому власний інтерфейс, бо сервер однаково відмовить.

На такій самій платформі приймання платежів і надання прав побудували як дві окремі системи. Картки через Stripe і покупки в застосунках Apple і Google сходяться в схемі payments із 9 таблиць і 34 функцій. Окрема схема access з 11 таблиць і 34 функцій відповідає на запитання «чи може цей обліковий запис це зробити?», не знаючи, яка вітрина заплатила. Від цієї відповіді залежить доступ до каталогу компаній, пошуку й експорту даних, і сервер перевіряє її заново під час кожного запиту, бо прихована кнопка — це люб’язність, а не засіб контролю. Нова вітрина тепер зачіпає лише платіжну частину, а відповідь на запит до підтримки про чийсь доступ шукають в одній таблиці замість трьох. Сторінка називає ціну: дві схеми й перевірку прав у запитах, які без неї нічого б не коштували.

Коли компанія перенесла вихідний код на власне обладнання, git-сервер налаштували так, щоб за замовчуванням він відмовляв. Сторінка зараховує контроль доступу до того, що означає тримати власний git-сервер. Його обрали за те, що він не має адміністративного вебінтерфейсу: що менше інтерфейсу, то менше треба захищати. Анонімний доступ заборонено, доступ без ключа заборонено, і кожен задекларований репозиторій

позначено як приватний, а не покладаються на те, що про нього ніхто не знає. Його SSH-сервер слухає лише loopback-інтерфейс на порту 23231, і до нього потрапляють через проміжний хост, а мультиплексор, який відкрив би git через публічний порт, написано й вимкнено, бо порт, яким ніхто не користується, — це теж поверхня атаки.

Де виникають проблеми

Перевірка живе в інтерфейсі. Кнопку приховано чи зроблено неактивною, а кінцева точка за нею відповідає кожному, хто звернеться до неї напряму. Сервер має ухвалити рішення ще раз під час кожного запиту, спираючись на факти, яких той, хто звертається, змінити не може.

Функцію перевіряють, а запис — ні. Запит називає рахунок, документ чи обліковий запис за його ідентифікатором, сервер підтверджує, що користувачеві можна переглядати рахунки, і повертає запитаний, не перевіривши, чий він. ASVS називає цю прогалину insecure direct object reference (IDOR), або broken object level authorization (BOLA). Захисник розпізнає її за ідентифікатором у запиті, який сервер використовує як є. Виправлення — шукати кожен об’єкт у межах того, до чого користувач має дозволений доступ, а в продукті з кількома клієнтами-орендарями брати орендаря із серверної сесії й ніколи із запиту.

Усе нове відкрите, доки хтось його не закриє. Новий маршрут, нова таблиця чи новий репозиторій починають життя з тим, що задано за замовчуванням, і якщо за замовчуванням усе дозволено, прогалина існує, доки хтось її не помітить. SP 800-53 формулює протилежне правило для мережевого трафіку як «заборонено за замовчуванням, дозволено як виняток», і це правило пасує кожному рівню: те, чого ніколи не надавали, заборонено.

Надані права тільки ростуть. SP 800-162 зазначає, що коли доступ із часом не прибирають і не відкликають, користувачі накопичують привілеї. Люди переходять в інші команди, проєкти завершуються, підрядники йдуть, і кожна така подія — момент, коли щось треба забрати. Відкликання, яке набуває чинності лише тоді, коли за кілька годин спливає сесія, — м’якша версія тієї самої прогалини.

Ніхто не може сказати, чому запит дозволили. Коли рішення ухвалюють у багатьох місцях і не журналюють ніде, розслідуванню нема чого читати. Записуйте дозволи й відмови разом із правилом, яке їх дало, і захищайте цей журнал: SP 800-162 попереджає, що за послідовністю рішень можна дізнатися атрибути людей, які стоять за запитами.

Що почитати далі

- Access control у глосарії NIST CSRC — визначення з FIPS 201-3 і CNSSI 4009, де авторизацію наведено як синонім.
- NIST SP 800-53 Rev. 5 — родина контролів доступу, зокрема AC-3 про втілення, AC-24 про рішення та AC-25.
- NIST SP 800-162 — контроль доступу на основі атрибутів із точками ухвалення й застосування рішень.
- OWASP Application Security Verification Standard — вимоги до авторизації, які можна перевірити, на рівні функцій, даних і полів.

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 контроль доступу?» 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/access-control/