Engineer

Cookie

невелике значення, яке сайт зберігає в браузері

Що це таке

Cookie, або HTTP cookie, у розмовній мові кукі, — це коротка пара з імені та значення, яку вебсервер надсилає браузеру в заголовку відповіді і яку браузер повертає в наступних запитах до того самого сервера. Саме cookies дають пам'ять протоколу HTTP, який зустрічає кожен запит як незнайомця: вхід, що витримує наступний клік, кошик, повний аж до оформлення замовлення, мову, яку обирають один раз.

Сервер встановлює кожен cookie окремим заголовком `Set-Cookie`, а браузер повертає всі доречні cookie одним заголовком `Cookie`. Тут перші три рядки — відповідь, а решта — пізніший запит із того самого браузера:

```
HTTP/1.1 200 OK
Set-Cookie: __Host-sid=4f2c9a71e0b8; Path=/; Secure; HttpOnly; SameSite=Lax
Set-Cookie: lang=da; Max-Age=31536000; Path=/

GET /account/ HTTP/1.1
Host: shop.example
Cookie: __Host-sid=4f2c9a71e0b8; lang=da
```

Усе після першої крапки з комою — атрибути, вказівки про те, скільки зберігати cookie і куди його надсилати. `Expires` або `Max-Age` задає найбільший строк життя; без жодного з них cookie живе, доки браузер не вирішить, що сесію завершено. `Domain` поширює cookie на весь домен і його піддомени, а без нього cookie повертається лише до хоста, який його встановив. `Path` звужує його до частини сайту. `Secure` не пускає його в незашифровані з'єднання, `HttpOnly` ховає його від скриптів, що читають `document.cookie`, а `SameSite` вирішує, чи їде він разом із запитами, які починає інший сайт.

Зворотний шлях несе лише імена й значення. Із заголовка `Cookie` сервер не дізнається, коли cookie спливає, чи встановили його з `Secure` і хто востаннє записав значення.

Правила описано в RFC 6265 2011 року. Його перегляд, відомий як rfc6265bis, додає `SameSite`, префікси імен `__Secure-` і `__Host-` та вимогу, щоб браузери обмежували будь-який строк життя щонайбільше 400 днями. На момент написання перегляд схвалено, і він чекає в черзі RFC Editor, щоб замінити RFC 6265.

Застосуйте поняття до рішення

Застосуйте «Що таке cookie?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Швидкий сайт, який легко знайти в пошуку і який ваші працівники редагують самі, — без рахунку за кожну зміну в тексті.

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. Розробила 100 вебзастосунків на HTML/HTML5 — 100 вебзастосунків побудовано

Розробила 100 вебзастосунків на HTML/HTML5 Зв'язатися з нами →

Чому це важливо

Cookie зазвичай єдине, що відрізняє запит користувача, який увійшов, від анонімного. Після входу ідентифікатор сесії в cookie для сервера і є користувачем: хто його пред’являє, той отримує обліковий запис, і пароля ніхто не питає. Більшість проблем із cookie зводиться до того, що хтось інший його читає, копіює чи використовує.

Останнє браузер робить легким. Він додає cookie до кожного запиту, що підпадає під його домен і шлях, хоч би хто цей запит почав. Форма на іншому сайті, що надсилає дані до інтернет-крамниці, приходить із cookie відвідувача, і сервер бачить запит від користувача, який увійшов. RFC 6265 називає це ambient authority і перелічує атаки, які це відкриває: підробку міжсайтових запитів (CSRF) і проблему «заплутаного заступника» (confused deputy).

Cookies — також найстаріший інструмент стеження в мережі. Скрипт, вбудований на тисячах сайтів, може встановити cookie від свого домену, який браузер повертатиме з кожного з цих сайтів, тож одна компанія може скласти докупи відвідини людини на всіх них. Перегляд описує таке

використання сторонніх cookie, зазначає, що більшість браузерів тепер блокує або розділяє їх, і радить політику настільки сувору, наскільки дозволяє сумісність.

Право стежить за зберіганням, а не за технологією. У ЄС стаття 5(3) Директиви про приватність і електронні комунікації (ePrivacy) дозволяє сайту зберігати інформацію на пристрої відвідувача або читати те, що там уже є, лише за згодою, наданою після ясної та вичерпної інформації. Винятки — зберігання лише для передавання повідомлення і зберігання, суворо необхідне для послуги, про яку відвідувач прямо попросив. Саме з цієї статті походять банери згоди, і вона взагалі не згадує cookie на ім'я. Настанови Європейської ради із захисту даних (EDPB) щодо її технічного обсягу застосовують її також до пікселів стеження і посилань для стеження.

Як це виглядає на практиці

Цей сайт їх не встановлює. Сторінка про cookies пояснює чому: немає облікових записів, кошика й причини стежити за відвідувачем, тож немає й банера згоди, бо погоджуватися нема на що. Вона також перелічує інші місця, де сайт може щось залишити, від `localStorage` до service worker, і каже, що їх теж не використано.

Відмова від cookie змінює те, як сайт дізнається про своїх читачів. Підрахунок читачів лише за журналом доступу вимірює читацьку аудиторію без кукі, скрипта чи пікселя, за власним журналом вебсервера, і відкидає заголовок `Cookie` з кожного рядка в момент запису, разом із заголовком авторизації. Результат навмисно грубий: унікальні відвідувачі та шлях відвідувача сайтом лишаються поза досяжністю, і сторінка друкує ці межі поряд із числами.

Там, де застосунку cookie таки потрібен, корисне питання — що йому довірено. На платформі, де одна людина діє від імені кількох організацій, активна організація зберігається в браузері, а серверний cookie дзеркалить її, щоб сторінки, які рендеряться на сервері, показували той самий контекст. Жодна з копій не вирішує доступу. Сервер перевіряє авторизацію на кожному запиті, тож той, хто змінить значення, змінить лише те, що показує його власний екран. Це безпечна форма для будь-якого cookie: значення може обирати, що показати, а рішення про те, що комусь дозволено, лишається на сервері.

Де виникають проблеми

Скрипт на сторінці читає сесію. Без `HttpOnly` скрипт, який вставив зловмисник, може прочитати `document.cookie` і надіслати ідентифікатор сесії куди завгодно. `HttpOnly` прибирає cookie з цього API. Це зупиняє крадіжку, але не зловживання: вставлений скрипт усе ще може надсилати запити, і браузер усе ще додає до них cookie, тож `HttpOnly` стоїть поряд із Content Security Policy, а не замість неї.

Cookie йде звичайним HTTP. Cookie без `Secure` вирушає з будь-яким запитом `http://` до свого хоста, і його може прочитати будь-хто на мережевому шляху. За RFC 6265 `Secure` захищає лише конфіденційність, і зловмисник у мережі все одно може перезаписати захищений cookie з незахищеного з'єднання. Перегляд звужує цю шпарину: ігнорує cookie з `Secure`, встановлені через звичайний HTTP, і відкидає cookie зі звичайного HTTP, який перекрив би захищений.

Сусідній піддомен встановлює cookie. `Domain=example.com` надсилає cookie кожному хосту в цьому домені, і будь-який із цих хостів може встановити cookie для всього домену, який головний сайт потім прочитає як свій. `Path` теж не є межею безпеки. Префікс `__Host-` відповідає на це: браузер приймає такий cookie, лише якщо він має `Secure` і `Path=/` та не має `Domain`, що прив'язує його до одного хоста, який його встановив.

Запит починає інший сайт. `SameSite=Strict` обмежує cookie запитами з того самого сайту, `Lax` додає переходи верхнього рівня безпечними методами, як-от перехід за посиланням, а `None` надсилає його всюди й ігнорується, якщо cookie не має також `Secure`. Перегляд поводиться з cookie без `SameSite` майже як із `Lax`; сервер, що на це розраховує, має вписати значення явно. Сайт у цьому сенсі — це схема плюс домен, який можна зареєструвати, тож `SameSite` нічого не дає проти ворожого сусіднього піддомену.

Сервер вірить значенню. Усе в cookie може змінити людина, у чиєму браузері він лежить. RFC 6265 радить серверам шифрувати й підписувати вміст і попереджає, що навіть тоді cookie можна відтворити повторно. Надійніший шаблон — випадковий ідентифікатор сесії, що вказує на стан, збережений на сервері, і замінюється під час входу, тож ідентифікатор, який зловмисник підкинув заздалегідь, — фіксація сесії, про яку попереджає RFC, — потім нічого не означає.

Браузер його відпускає. Дата завершення — це максимум. Браузери видаляють cookie, щоб не виходити за свої ліміти, і зловмисник може витіснити cookie, встановивши багато власних. Серверам не варто розраховувати, що браузери збережуть cookie. Видалення теж має пастку: `Set-Cookie` з датою в минулому видаляє cookie, лише якщо його `Path` і `Domain` збігаються з тими, з якими його створили.

Що почитати далі

- RFC 6265: HTTP State Management Mechanism — стандарт для `Set-Cookie` і `Cookie` з правилами атрибутів і розділом про безпеку: ambient authority, слабка цілісність та ідентифікатори сесій.
- Cookies: HTTP State Management Mechanism (rfc6265bis) — схвалений перегляд із `SameSite`, префіксами `__Secure-` і `__Host-`, обмеженням у 400 днів і розділом про сторонні cookie.
- Директива 2002/58/ЕС, консолідований текст — англійською: директива ePrivacy зі змінами 2009 року і статтею 5(3) про зберігання й читання інформації на пристрої користувача.
- Настанови EDPB 2/2023 щодо технічного обсягу статті 5(3) — англійською: як правило поширюється на пікселі стеження, посилання для стеження, локальну обробку та інші техніки, крім cookie.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке cookie?» 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/cookie/