Engineer

Протокол

узгоджені правила, за якими спілкуються дві програми

Що це таке

Протокол — це письмова домовленість про розмову між програмами: який вигляд має кожне повідомлення, що воно означає, яка сторона говорить першою, що може йти після чого і що робить кожна сторона, коли інша каже щось неочікуване. Програми, написані різними людьми різними мовами, можуть працювати разом, бо кожну збудовано за тим самим документом, а не одна під одну.

Глосарій NIST збирає кілька визначень з публікацій інституту, і в них є спільне ядро: правила щодо формату повідомлень і процедур навколо них, яких дотримуються дві чи більше сторін, щоб досягти спільної мети. Одне з визначень називає половину, яку легко забути: правила описують порядок повідомлень так само, як і їхню будову. Повідомлення в правильному форматі, надіслане в неправильний момент, ламає розмову так само надійно, як і зіпсоване.

Специфікації, опубліковані як RFC, формулюють свої правила ключовими словами великими літерами. MUST — це абсолютна вимога. SHOULD дозволяє відступ лише тоді, коли всі наслідки зрозуміли й зважили. MAY позначає щось справді необов'язкове. З 2017 року ці слова мають таке значення лише великими літерами, тож «must» малими літерами в тому самому документі — звичайна англійська. Відстань між SHOULD і MUST — це відстань між програмою, яка поводиться незвично, і програмою, яка помиляється.

Протоколи складаються в шари. Прикладний протокол, тобто та частина, що знає про сторінки, меню чи пошту, зазвичай працює поверх транспортного протоколу, єдине завдання якого — переносити байти між двома машинами. TCP, транспорт, яким користується більшість із них, за словами своєї

специфікації надає надійний потік байтів, що зберігає порядок: байти приходять повністю й у тому порядку, в якому їх надіслали. Водночас він не обіцяє жодного зв’язку між шматками, які записує одна сторона, і шматками, які читає інша. У потоці немає повідомлень, тож кожен протокол над ним мусить сам позначати, де закінчуються його повідомлення: кінцем рядка, довжиною, надісланою наперед, закриттям з’єднання або іншою домовленою позначкою. UDP укладає протилежну угоду. Він переносить окремі датаграми з мінімумом механізмів, і ні доставку, ні захист від дублікатів не гарантовано. Кожен транспорт розрізняє служби на одній машині за номером порту, і ці два транспорти нумерують свої порти окремо.

Решта характеру протоколу випливає з кількох рішень, які ухвалює кожен дизайн, свідомо чи ні:

- **Хто говорить першим.** Одні сервери вітають клієнта, інші мовчки чекають на запит, а коли чекають обидві сторони, нічого не відбувається, доки не спливе таймер.
- **Що пам’ятає сервер.** У протоколі без стану кожен запит несе все, що потрібно, щоб на нього відповісти. У протоколі зі станом значення повідомлення залежить від того, що було перед ним у тому самому з’єднанні.
- **Як він росте.** Номери версій, коди опцій і поля, які треба ігнорувати, якщо вони невідомі, — це місце, яке протокол залишає для власного майбутнього.

Застосуйте поняття до рішення

Застосуйте «Що таке протокол?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Wi-Fi, фаєрволи і віддалений доступ налаштовані так, щоб працювати можна було звідусіль, а заходити — лише своїм.

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. Керувала 30 інстансами Ubuntu Linux VPS — 30 серверів Linux під управлінням, із планом відновлення

Керувала 30 інстансами Ubuntu Linux VPS Зв'язатися з нами →

Чому це важливо

Протокол — це те, що робить замінним кожен кінець. Будь-який клієнт, написаний за документом, може говорити з будь-яким сервером, написаним за ним, тож читач може змінити програму, а сервер можна переписати

іншою мовою, і інша сторона цього не помітить. Програми, які узгоджені лише тому, що їх створювали разом, не можуть змінюватися незалежно одна від одної.

Протокол також визначає значну частину безпеки ще до того, як з'явиться хоч рядок коду. Те, яка сторона відкриває з'єднання, вирішує, що треба впускати: серверу, який чекає на клієнтів, потрібен порт, відкритий у світ, а програмі, яка лише сама виходить назовні, порт не потрібен. Те, що несуть повідомлення, вирішує, що можна записати в журнал, порахувати або що може витекти. Запит, у якому вказано шлях, можна записати в журнал за шляхом, а приховування мережевої адреси читача не заважає серверу підраховувати запити. А протокол без шифрування показує кожен запит кожній мережі, через яку проходить.

І нарешті, протоколи переживають програми, які ними говорять. Реалізації замінюють кожні кілька років, а розмова триває десятиліттями, і будь-яка поведінка, на яку покладаються розгорнуті програми, стає її частиною, хоч би що казав документ. Рада з архітектури інтернету (Internet Architecture Board) писала про обидві половини цієї проблеми: про помилки, які терплять, доки вони не стануть стандартом, і про місце для росту, яким ніхто не користується, доки воно не перестане працювати.

Як це виглядає на практиці

Цей сайт доступний за чотирма протоколами — з однієї теки файлів на одному сервері: HTTP, Gemini, Gopher і onion-сервіс у мережі Tor. Сторінка описує Gopher як протокол меню 1991 року — простий TCP, без шифрування, без скриптів — і наводить цілий запит, набраний вручну:

```
printf '/uk/contact\r\n' | nc engineer.company 70
```

Вона каже, що в цьому одному рядку весь протокол. Під ним одне розгортання записує одну теку, а чотири входи читають її, тож жоден із них не є окремо підтримуваною копією сайту.

Правила брандмауера перевіряють за позицією, а не лише за наявністю, бо фільтр пакетів перебирає впорядкований список і зупиняється на першому збігу. Одне з правил перевірки стосується протоколів: на кожен порт — рівно

одне правило, і завжди із зазначеним протоколом, бо правило без протоколу непомітно розширює поверхню атаки. Оголошена публічна поверхня — це короткий перелік із п'яти портів, який звіряють цілком, тож порт, що з'явився без оголошення, провалює перевірку.

Четвертий вхід — це onion-дзеркало, і його протокол визначив і брандмауер, і облік відвідувань. Демон анонімності підключається до мережі, будуючи вихідні ланцюги, тож ззовні ніщо не підключається, жоден порт не відкривається, і жодному з двох рівнів брандмауера нічого синхронізувати. Запити Gemini та з'єднання Gopher підраховують за кожен день без адреси й без шляху. Для onion немає ні лічильника, ні прапорця, щоб його додати, бо цей сайт свідомо не підраховує цю аудиторію.

Де виникають проблеми

Помилка, яку терплять, стає стандартом. Принцип надійності — будь консервативним у тому, що надсилаєш, і ліберальним у тому, що приймаєш, — мав допомогти програмам працювати разом. Застосований без меж, він дає змогу хибній реалізації вийти й поширитися. Наступна реалізація мусить приймати ту саму ваду, щоб працювати з нею, і вада стає стандартом де-факто, з яким усе нове має бути сумісним аж до помилок. RFC 9413 каже про це прямо: розгорнуті реалізації стають тестом на відповідність, який може переважити специфікацію. Відхиляйте те, що специфікація забороняє, якомога раніше в житті протоколу, і робіть помилку видимою для того, хто може її виправити.

Дві програми не погоджуються, де закінчується повідомлення. У потоці байтів межа там, де її ставить протокол. Протокол, який пропонує два способи її позначити, або парсер, що приймає недбалий варіант, дає проксі та серверу за ним прочитати ті самі байти як різні повідомлення. Те, що один вважає хвостом запиту, інший вважає новим запитом, якого проксі ніколи не перевіряв. Дозвольте одне правило розмежування, відхиляйте все неоднозначне й зробіть так, щоб кожна ланка розбирала його однаково.

Правило називає порт, але не протокол. Номер порту нічого не означає, доки не названо транспорт, і залежно від інструмента правило брандмауера, записане лише з числом, може відкрити порт і для TCP, і для UDP. Транспорт, який не був потрібен, тоді відкритий для всього, що згодом там слухатиме. Вписуйте протокол у кожне правило й звіряйте всю відкриту поверхню з оголошеним переліком.

Місце для росту перестає працювати. Якщо жоден розгорнутий відправник ніколи не користується точками розширення протоколу, з’являються отримувачі, які відкидають усе, чого ніколи не бачили, і перше справжнє розширення ламається об них. Що довше працює такий отримувач, то важче його виправити. Використовуйте точки розширення від самого початку, зокрема зі значеннями, які нічого не означають, щоб терпимість перевірялася постійно, а не бралася на віру.

Що почитати далі

- Глосарій NIST CSRC: protocol — визначення з публікацій NIST з безпеки, кожне з документом, звідки воно походить.
- RFC 9293: Transmission Control Protocol — чинний стандарт TCP, зокрема потік байтів без меж між повідомленнями.
- RFC 768: User Datagram Protocol — увесь UDP на трьох сторінках, 1980 рік.
- RFC 1436: The Internet Gopher Protocol — інформаційний опис цілого прикладного протоколу з прикладами сеансів.
- RFC 9110: HTTP Semantics — значення запитів, відповідей, методів і кодів стану незалежно від версії, що йде мережею.
- RFC 9413: Maintaining Robust Protocols — Рада з архітектури інтернету (IAB) про те, чому терпимість до помилок закріплює їх.

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 протокол?» 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/protocol/