Engineer

CIDR

⊞ запис і маршрутизація блоків адрес довільного розміру

Що це таке

CIDR (англійською Classless Inter-Domain Routing, безкласова міждоменна маршрутизація) — це спосіб описати блок IP-адрес початковою адресою та кількістю старших бітів, спільних для всіх адрес блоку. На письмі це знайома скісна риска: 198.51.100.0/24 означає всі адреси, перші 24 біти яких збігаються з 198.51.100.0, тобто 256 адрес від 198.51.100.0 до 198.51.100.255. Число після риски називають довжиною префікса, а сам блок — префіксом.

Слово «безкласова» вказує на те, що було раніше. Початковий план IPv4 з RFC 791 передбачав мережі трьох розмірів, і RFC 4632, чинний опис CIDR, пояснює, чому це не спрацювало: клас C, щонайбільше 254 адреси хостів, був замалий для середньої організації, а клас B, до 65 534 адрес, — завеликий, проте саме він підходив найкраще з того, що було. Коли наприкінці 1980-х мережа стала комерційною, попит на клас B зростав, як здавалося, експоненційно.

У січні 1992 року група ROAD (Routing and Addressing) в IETF назвала три проблеми: вичерпання простору класу B, зростання таблиць маршрутизації понад те, з чим могли впоратися маршрутизатори та їхні оператори, і зрештою вичерпання адрес самого IPv4. Перші дві мали стати критичними десь між 1993 і 1995 роками. CIDR став відповіддю саме на них: його вперше запропонували в RFC 1338 у червні 1992 року, потім у RFC 1519 у вересні 1993 року, а в серпні 2006 року RFC 1519 замінив RFC 4632. Третю проблему CIDR ніколи не мав розв’язати, лише виграти час.

У записі CIDR, за RFC 4632, префікс складається зі звичної адреси з чотирьох частин, скісної риски й числа від 0 до 32, яке задає кількість значущих бітів. Старіша маска мережі каже те саме: ці біти в ній одиниці, решта — нулі, тож /24 — це 255.255.255.0. Єдине обмеження, яке зберігає RFC, — маска має

бути неперервною, і в цих межах префікс може описати будь-який блок, розмір якого є степенем двійки. Таблиця в RFC іде від /32, маршруту до хоста з однією адресою, через /31 для з'єднання «точка — точка» з двома адресами, /24 з 256 і /16 з 65 536 — аж до 0.0.0.0/0, усіх 4 294 967 296 адрес, що слугує маршрутом за замовчуванням.

Запис пережив сімейство адрес, для якого його створили: RFC 4291, що визначає будову адрес IPv6, записує префікси IPv6 так само, тож 2001:db8::/32 читається точнісінько як його родичі з IPv4.

Застосуйте поняття до рішення

Застосуйте «Що таке CIDR?» 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

CIDR розв'язав свої початкові проблеми агрегацією. RFC 4632 називає агрегацію інформації єдиним загальновідомим способом зменшити стан маршрутизації в мережі з комутацією пакетів. Провайдер, що має один великий блок і нумерує клієнтів його невеликими частинами, може анонсувати один маршрут для всього блоку, і решті інтернету більше нічого не треба, щоб дістатися кожного клієнта в ньому. RFC 4632 додає, що для цього довелося змінити сам порядок розподілу адрес і що частина старого простору класу C, яку не можна агрегувати, англійською the swamp, «болото», у 2006 році, за оцінкою, все ще займала близько 15 000 записів у таблиці маршрутизації.

Агрегація працює завдяки одному правилу, якого дотримуються всі маршрутизатори. За словами RFC 4632: «Пересилання в інтернеті здійснюється за принципом найдовшого збігу». Коли кілька префіксів охоплюють одне призначення, перемагає найконкретніший. Так організація, що змінює провайдера без перенумерації, лишається досяжною: новий провайдер анонсує її власний невеликий префікс — RFC описує це як

пробивання діри в агрегаті старого провайдера, — і йому віддають перевагу перед агрегатом, бо це довший збіг. CIDR планували приблизно на три–п’ять років, а за словами RFC, до 2005 року його було повністю впроваджено.

Мало хто з інженерів налаштовує міждоменну маршрутизацію, але майже всі пишуть префікси: у правилі фаєрвола, що впускає мережу офісу, у таблиці маршрутів на Linux-сервері та у WireGuard, де список префіксів AllowedIPs для кожного вузла визначає і те, з яких адрес-джерел приймати трафік від цього вузла, і те, куди надсилати трафік для нього. Скрізь одна хибна цифра після риски непомітно впускає або відсікає тисячі хостів.

Як це виглядає на практиці

Стандартний модуль Python `ipaddress` розуміє цей запис і дає змогу швидко перевірити арифметику, перш ніж вона потрапить у правило. Сеанс нижче — ілюстрація цієї статті, виконана на Python 3.15 з адресами з документаційного діапазону 198.51.100.0/24:

```
>>> import ipaddress
>>> net = ipaddress.ip_network("198.51.100.0/24")
>>> net.netmask, net.num_addresses
(IPv4Address('255.255.255.0'), 256)
>>> for sub in net.subnets(new_prefix=26):
...     print(sub)
...
198.51.100.0/26
198.51.100.64/26
198.51.100.128/26
198.51.100.192/26
>>> ipaddress.ip_network("198.51.100.70/26")
Traceback (most recent call last):
  ...
ValueError: 198.51.100.70/26 has host bits set
>>> ipaddress.ip_network("198.51.100.70/26", strict=False)
IPv4Network('198.51.100.64/26')
>>> half = ipaddress.ip_network("198.51.100.128/25")
>>> ipaddress.ip_address("198.51.100.200") in half
True
>>> half.subnet_of(net)
True
```

/24 ділиться на чотири блоки /26 по 64 адреси, і кожен починається з числа, кратного 64. Спроба створити 198.51.100.70/26 завершується помилкою, бо блок, що містить .70, починається з .64, а біти після 26-го не нульові; з

`strict=False` модуль обнуляє їх і повертає блок, до якого належить .70. Останні два рядки — це питання, які варто ставити до кожної пари префіксів під час рецензування правил: чи потрапляє адреса всередину блоку і чи лежить один блок цілком усередині іншого.

Друге питання найважливіше для фільтра пакетів, бо він не користується найдовшим збігом. Він читає правила по порядку й зупиняється на першому, яке ухвалює рішення. Посібник nft(8) каже, що `accept` і `drop` завершують обробку ланцюжка, а `add` додає правило в кінець ланцюжка, тоді як `insert` ставить його на початок. У другій ілюстрації цієї статті мета — дозволити блоку /24 доступ до SSH, але не пустити його верхню половину:

```
add table inet filter
add chain inet filter input { type filter hook input priority filter; }
add rule inet filter input ip saddr 198.51.100.0/24 tcp dport 22 accept
add rule inet filter input ip saddr 198.51.100.128/25 tcp dport 22 drop
```

Обидва правила виглядають правильно, і в переліку видно обидва. Але кожна адреса з /25 є і в /24, тож перше правило приймає трафік звідти, а правило drop під ним ніколи не спрацьовує. У маршрутизації конкретніший префікс переміг би, хоч би де він стояв; у фільтрі він перемагає, лише якщо стоїть першим. Якщо поміняти ці два рядки місцями або додати вужче правило через `insert`, вийде задуманий результат. `nft -a list ruleset` виводить кожне правило з його дескриптором (handle), а `add` та `insert` приймають дескриптор, щоб поставити нове правило поруч із наявним.

Запис у портфоліо Перевірила правила фаєрвола за позицією за серпень 2026 року описує ту саму пастку з боку експлуатації. Хост цілий рік працював з обмеженням частоти, до якого не дійшов жоден пакет, бо воно стояло нижче ширшого правила, яке спрацьовувало першим, і кожен причетний інструмент увесь цей час правильно виводив це обмеження. Тепер перевірки читають нумерований перелік правил і живий ланцюжок фільтра та звіряють саме порядок. Сторінка не каже, як було записано ширше правило і чи йшлося про префікси адрес, тож рядки вище ілюструють принцип, а не правила того хоста.

Де виникають проблеми

Біти хоста в адресі. 198.51.100.70/26 — це адреса з маскою, а не блок. Python за замовчуванням її відхиляє, але з `strict=False` округлює вниз до 198.51.100.64/26, а це можуть бути зовсім не ті хости, які мав на увазі автор. Пишіть адресу мережі блоку й розбирайте записи в строгому режимі, щоб недогляд проявився як повідомлення про помилку.

Одна адреса там, де мав бути блок. Документація Python каже, що мережу без маски сприймають як /32, тож адреса офісу, записана без префікса, впускає один хост. Завжди пишіть довжину префікса, навіть /32, щоб читач міг відрізнити навмисний маршрут до хоста від забутої риски.

Широкий префікс над вузьким. Як показує ілюстрація, у фільтрі за першим збігом попереднє правило затінює кожне пізніше, чий префікс воно містить. Ставте найконкретніші правила першими, а коли правила генеруються, перевіряйте кожну пару на вкладеність, наприклад через `subnet_of` або `overlaps`, і зупиняйте збірку, якщо пізніше правило недосяжне.

Довжина префікса, коротша на кілька бітів. Кожен прибраний біт подвоює блок, тож /16, записаний замість /24, впускає в 256 разів більше адрес, а /0 — усі адреси, які існують. RFC 4632 фіксує, що через помилки операторів та інші події в минулому в деяких мережах спостерігалися маршрути /1 і /2. Вважайте кожне правило, коротше за блок, яким ви володієте, приводом для рецензії, а 0.0.0.0/0 і ::/0 залишайте для правил, які мають збігатися з усім.

Хости, пораховані як адреси. У /24 256 адрес, але у звичайній мережі перша з них — адреса мережі, а остання — широкомовна, тож `hosts()` у Python повертає 254 з них. На з’єднаннях «точка — точка» ці втрати накопичуються: RFC 3021 зазначає, що /30 витрачає чотири адреси на два хости, і дозволяє таким з’єднанням використовувати /31, що заощаджує 1000 адрес на 500 з’єднаннях. Обирайте розмір блоку за кількістю хостів, які він має вмістити, плюс зарезервовані адреси, а /31 використовуйте лише на пристроях, що підтримують RFC 3021.

Агрегат без маршруту discard. Маршрутизатор, що анонсує зведений блок, але втратив один із менших префіксів усередині нього, може відправити трафік для відсутньої частини назад менш конкретним маршрутом, наприклад маршрутом за замовчуванням, і так створити петлю пересилання.

RFC 4632 вимагає, щоб маршрутизатор, який утворює агрегат, відкидав пакети, що збігаються з ним, але не збігаються з жодним конкретнішим маршрутом, а на практиці це означає встановити маршрут discard або null для кожного агрегату, який він анонсує.

Списки дозволів лише для IPv4. У nftables `ip saddr` читає заголовок IPv4, а `ip6 saddr` — заголовок IPv6, а в Python спроба згорнути змішаний список дає `TypeError`, бо дві версії не можна поєднати. Список префіксів IPv4 нічого не каже про клієнтів з IPv6, які тоді провалюються до того, що стоїть далі. Тримайте префікси IPv4 та IPv6 поруч у кожному правилі й перевіряйте обидва під час тестування.

Що почитати далі

- RFC 4632: Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan — чинний опис CIDR з історією, таблицею префіксів і правилами агрегації.
- RFC 3021: Using 31-Bit Prefixes on IPv4 Point-to-Point Links — чому /30 витрачає половину адрес на з’єднання «точка — точка» і як /31 цього уникає.
- RFC 4291: IP Version 6 Addressing Architecture — правила адресації IPv6, зокрема префікси в тому самому записі зі скісною рискою.
- ipaddress: IPv4/IPv6 manipulation library — довідка Python про розбір, поділ, порівняння та згортання префіксів.
- nft(8): nftables manual — довідник nftables, зокрема про порядок правил і рішення, що завершують ланцюжок.
- wg(8): set and retrieve configuration of WireGuard interfaces — посібник WireGuard, де AllowedIPs визначено як список адрес із масками CIDR.

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 CIDR?» 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/cidr/