Документація до програми firewall.py

1) Система складається з чотирьох взаємопов'язаних модулів:

RuleEngine - центральний рушій обробки правил. Цей компонент зберігає всі активні правила фільтрації, застосовує їх за пріоритетом, виявляє конфлікти та дублікати. Кожне нове правило автоматично відсортовується за пріоритетом (менше число = вищий пріоритет), що забезпечує передбачувану послідовність обробки. Для кожного вхідного пакету чи з'єднання RuleEngine здійснює послідовний пошук першого збіглого правила та повертає дію (allow або deny).

AdminAPI - REST API на базі Flask, що надає інтерфейс управління правилами в реальному часі. Користувач або автоматизований сценарій можуть отримати список всіх правил (GET /rules), додати нове правило (POST /rules), змінити існуюче (PATCH /rules/{id}), видалити його (DELETE /rules/{id}) або перезавантажити конфігурацію (POST /reload). Усі операції виконуються без перезапуску основного процесу - це дозволяє гарячу зміну правил.

EventLogger - модуль журналювання, що записує кожну подію блокування або дозволу трафіку у файл у форматі JSONL (одна подія = один рядок JSON). Кожний запис містить часову мітку, тип дії, протокол, адреси джерела та призначення, номер порту та причину (ідентифікатор збіглого правила або повідомлення про причину). Цей журнал служить основою для подальшого аналізу, пошуку аномалій та аудиту.

TCP Proxy - демонстраційний проксі-сервер, що прослуховує певний локальний порт і переспрямовує трафік на віддалений хост/порт. На кожне нове з'єднання RuleEngine застосовує правила; якщо дія - deny, з'єднання одразу закривається й подія логується. Якщо дія - allow, встановлюється туннель між клієнтом та цільовим сервером.

2) Структура та формальний опис правил

Описання полів правила

Кожне правило фільтрації має такі поля:

id – унікальний ідентифікатор правила (тип: число). Програма автоматично присвоює id при створенні. Приклади: 1, 2, 3.

action - дія, яка повинна бути виконана, якщо пакет збігається з правилом (тип: рядок). Дозволені значення: "allow" (дозволити пакет) або "deny" (блокувати пакет).

src_ip_cidr - IP-адреса або CIDR-діапазон джерела пакету (тип: рядок). Це визначає, звідки може прибути трафік. Приклади: "192.168.1.0/24" (усі IP від 192.168.1.0 до 192.168.1.255), "10.0.0.5" (конкретна адреса), "0.0.0.0/0" (будь-яка адреса).

dst_ip_cidr - IP-адреса або CIDR-діапазон призначення пакету (тип: рядок). Це визначає, куди спрямований трафік. Приклади: "0.0.0.0/0" (будь-яка адреса), "8.8.8.0/24" (Google DNS).

src_port - порт джерела (тип: число). Якщо значення = 0, це означає будь-який порт. Приклади: 0 (будь-який), 1024 (конкретний порт), 55555.

dst_port - порт призначення (тип: число). Якщо значення = 0, це означає будь-який порт. Приклади: 80 (HTTP), 443 (HTTPS), 8080 (локальна послуга), 0 (будь-який).

proto - протокол мережі (тип: рядок). Дозволені значення: "TCP" або "UDP".

priority - пріоритет обробки правила (тип: число). Менше число = вище пріоритет. Правила обробляються у порядку зростання пріоритету. Приклади: 1 (найвищий пріоритет), 10, 100 (найнижчий пріоритет).

enabled - статус активності правила (тип: істина/неправда). Якщо значення = true, правило працює. Якщо значення = false, правило ігнорується. Приклади: true (активне), false (відключене).

3) Функціональність: Що програма виконує

1. Динамічна фільтрація трафіку

Програма прослуховує на локальному порті (за замовчуванням - TCP порт 12345) і переспрямовує вхідні з'єднання на цільовий сервер. Перед встановленням з'єднання RuleEngine аналізує правила:

Вхідне з'єднання від 192.168.1.100 на порту 55000, яке намагається досягти 127.0.0.1 на порту 12345 поступає до RuleEngine.

RuleEngine звертається до всіх активних правил у порядку пріоритету та шукає перше, яке збігається з цим з'єднанням.

Якщо перше збіглене правило має action=deny, то z'єднання закривається одразу, подія записується до журналу, процес припиняється.

Якщо action=allow, встановлюється туннель (проксі-з'єднання) до цільового хоста (target_host:target_port), і трафік починає передаватися туди.

2. Правила без перезапуску (Hot reload)

За допомогою REST API можна в будь-який момент додавати, видаляти або змінювати правила без вимкнення та запуску брандмауера. Це досягається за допомогою блокування та атомарних операцій. Таким чином, адміністратор може оновлювати правила навіть під час активного трафіку.

3. Журналювання всіх подій

Кожен факт блокування або дозволу записується в JSONL-файл fw_events.log.jsonl. JSONL означає, що кожен рядок файлу - це окремий JSON-об'єкт. Це робить файл легким для аналізу комп'ютером.

4. Виявлення конфліктів та дублікатів

При додаванні нового правила RuleEngine автоматично перевіряє наявність проблем.

Точні дублікати - це коли в системі вже існує правило з однаковими полями src_ip_cidr, dst_ip_cidr, src_port, dst_port, proto та однаковою дією. Якщо хтось спробує додати такий же запис ще раз, система повертає попередження типу: Duplicate of rule 2. Це означає, що новоствореного правила 5 є дублікатом правила 2.

Конфлікти дій - це коли два правила мають однакові умови фільтрації (той же IP, порти, протокол), але різні дії. Наприклад, перше правило говорить "блокувати TCP на порт 80", а друге говорить "дозволити TCP на порт 80". Це логічна помилка, адже система не може одночасно дозволити й заблокувати. Однак через пріоритети, перше правило все одно переможе.

Система записує ці конфлікти у відповідь API, дещо типу: Conflict with rule 1 (deny vs allow). Це повідомлює адміністратора про потенційну помилку.

Важливо: система дозволяє правилу бути доданим навіть якщо є конфлікти. Адміністратор повинен самостійно розібратися та видалити невірне правило.

5. Пріоритети та послідовність обробки

Правила сортуються за priority при кожному додаванні або зміні. Це означає, що система завжди знає, в якому порядку обробляти правила.

Приклад ситуації з пріоритетами:

У вас є два правила:

Правило 1: priority=10, action=deny, dst_port=80

Правило 2: priority=5, action=allow, dst_port=80

Оскільки Правило 2 має priority=5, а Правило 1 має priority=10, Правило 2 обробиться першим (5 менше за 10). Тому весь трафік на port 80 буде дозволений, незважаючи на те, що Правило 1 каже його блокувати.

Це дозволяє комбінувати дозволи та блокування за допомогою пріоритетів:

priority=1 означає критичні блокування (найвище), вони перевіряються в першу чергу.

priority=5 означає загальні дозволи для своєї мережі.

priority=100 означає fallback-правила (найнижче), вони перевіряються в останню чергу.