ТЕХНІЧНЕ ЗАВДАННЯ

Тема: Розробка програмного забезпечення для розгортки та встановлення безпечного vpn-з’єднання і віддаленого доступу до пристрою.

Мета проєкту: Розробити комплексне програмне рішення, що дозволяє користувачам легко створювати приватні VPN-мережі, керувати доступом користувачів та груп, і автоматично отримувати необхідні конфігураційні файли для підключення. Надати можливість вбудованого віддаленого доступу через термінальну оболонку. Мета – абстрагуватися від складності ручного налаштування OpenVPN та PKI (Public Key Infrastructure), надати можливість віддаленого доступу.

Вимоги:

1. Платформа та сумісність:
   a. Серверна частина програми повинна бути розроблена для роботи на 64-бітних версіях ОС Linux (розробка ведеться на Arch). У подальшому можна перенести все на Docker.
   b. Клієнтська частина повинна бути кросплатформенною, з пріоритетною підтримкою 64-бітних ОС Linux та можливістю компіляції та запуску під ОС Windows.
2. Структура:
   a. “Сервер” реально являє собою декілька серверів та клієнтів:
      1. VPN-сервер
      2. SMTP-клієнт
      3. HTTPS-сервер
      4. Database-клієнт
   b. Клієнт реально являяє собою:
      1. VPN-клієнт
      2. SSL/STL-клієнт
   c. “Common libs”: ”частини коду”, що використовуватимуться, як на серверній так і клієнтській частині.
3. Технологічний стек:
   a. Програма повинна бути написана переважно на мові програмування C++ (стандарт 17)
   b. Серверна частина повинна використовувати бібліотеку Boost.Asio та Boost.Beast для асинхронної реалізації HTTPS-сервера, libcurl для SMTP – клієнта.
   c. Клієнтська частина повинна використовувати Qt6 для написання графічної частини та, можливо, збереження токенів (небезпечно, але підходить для старту).
   d. Для криптографічних операцій (хешування, генерація токенів) повинна використовуватись бібліотека libsodium.
   e. Для зберігання даних – Sqlite3. (Клієнт для використання БД написати з врахуванням можливої міграції до більш продвинутих БД)
   f. Збірка залежностей – Cmake, білд – Ninja.
   g. Реалізація віддаленого доступу до пристрою через термінал – libssh

h. Обробка медіа (кодування, декодування, захоплення): Бібліотеки FFmpeg (libavcodec, libavformat, libavdevice, libswscale).
i. Потокова передача в реальному часі SFU (Selective Forwarding unit):
    1. Janus Gateway;
    2. LiveKit.

4. Безпека та мережа:
    a. Програма повинна забезпечувати створення безпечного VPN-з'єднання на базі OpenVPN
    b. Увесь управляючий трафік між клієнтом та сервером (реєстрація, авторизація, передача конфігурацій) має бути захищений за допомогою шифрування TLS.
    c. Програма повинна реалізувати систему автентифікації користувачів за логіном (email) та паролем, а також систему сесій на основі Access та Refresh токенів
    d. Паролі користувачів повинні зберігатися у вигляді захищених хешів (з використанням солі та алгоритму Argon2id13).
5. Функціональність сервера:
    a. Сервер повинен автоматично генерувати унікальні конфігураційні файли (.ovpn) та сертифікати для кожного зареєстрованого та верифікованого користувача.
    b. Сервер повинен мати можливість створювати ізольовані групи користувачів та динамічно керувати мережевими правилами доступу між ними за допомогою nftables.
    c. Сервер повинен підтримувати механізм верифікації користувачів через підтвердження електронної пошти.
6. Функціональність клієнта:
    a. Програма повинна мати графічний інтерфейс користувача (GUI), реалізований на Qt.
    b. Клієнт повинен надавати інтерфейс для реєстрації, входу та (опціонально) відновлення пароля.
    c. Клієнт повинен автоматично отримувати та зберігати конфігураційний файл VPN після успішної авторизації.
    d. Клієнт повинен мати можливість ініціювати та розривати VPNз'єднання, взаємодіючи з локально встановленим процесом OpenVPN.
    e. Клієнт повинен відображати список груп, до яких належить користувач, та надавати інтерфейс для управління групами (якщо користувач є їх власником).
    f. Клієнт має реалізовувати ssh-клієнт/сервер для реалізації віддаленого доступу.
    g. Додається новий тип сесії "Віддалений доступ з відеотрансляцією".
    h. Клієнт (що транслює): При старті сесії починає захоплення екрана (через libavdevice), кодує відео в H.264 (через libavcodec) і відправляє його через обраний протокол (WebRTC або RTP).

i. Клієнт (що переглядає): Приймає потік даних, декодує його (через libavcodec) і відображає у спеціальному віджеті в Qt-інтерфейсі (наприклад, QOpenGLWidget для максимальної продуктивності).

Мінімальний життєздатний продукт

Мета MVP: Створити функціональне ядро системи, яке дозволяє користувачам реєструватися, отримувати персональні конфігураційні файли OpenVPN та керувати VPN-з'єднанням через клієнтський GUI-додаток.

Функціональні вимоги MVP:

1. Платформа та сумісність:
   a. Серверна частина програми повинна бути розроблена для роботи на 64-бітних версіях ОС Linux.
   b. Клієнтська частина повинна бути кросплатформенною, з пріоритетною підтримкою 64-бітних ОС Linux та можливістю компіляції та запуску під ОС Windows.
2. Структура:
   a. “Сервер” реально являє собою декілька серверів та клієнтів:
      1. VPN-сервер
      2. SMTP-клієнт
      3. HTTPS-сервер
      4. Database-клієнт
   b. Клієнт реально являяє собою:
      1. VPN-клієнт
      2. SSL/STL-клієнт
   c. “Common libs”: ”частини коду”, що використовуватимуться, як на серверній так і клієнтській частині.
3. Технологічний стек:
   a. Програма повинна бути написана переважно на мові програмування C++ (стандарт 17)
   b. Серверна частина повинна використовувати бібліотеку Boost.Asio та Boost.Beast для асинхронної реалізації HTTPS-сервера, libcurl для SMTP – клієнта.
   c. Клієнтська частина повинна використовувати Qt6 для написання графічної частини та, можливо, збереження токенів.
   d. Для криптографічних операцій (хешування, генерація токенів) повинна використовуватись бібліотека libsodium.
   e. Для зберігання даних – Sqlite3.
   f. Збірка залежностей – Cmake, білд – Ninja.
   g. Реалізація віддаленого доступу до пристрою через термінал – libssh
4. Безпека та мережа:
   a. Програма повинна забезпечувати створення безпечного VPN-з'єднання на базі OpenVPN
   b. Увесь управляючий трафік між клієнтом та сервером (реєстрація, авторизація, передача конфігурацій) має бути захищений за допомогою шифрування TLS.

c. Програма повинна реалізувати систему автентифікації користувачів за логіном (email) та паролем, а також систему сесій на основі Access та Refresh токенів
d. Паролі користувачів повинні зберігатися у вигляді захищених хешів (з використанням солі та алгоритму Argon2id13).

5. Функціональність сервера:
   a. Сервер повинен автоматично генерувати унікальні конфігураційні файли (.ovpn) та сертифікати для кожного зареєстрованого та верифікованого користувача.
   b. Сервер повинен мати можливість створювати ізольовані групи користувачів та динамічно керувати мережевими правилами доступу між ними за допомогою nftables.
   c. Сервер повинен підтримувати механізм верифікації користувачів через підтвердження електронної пошти.
6. Функціональність клієнта:
   a. Програма повинна мати графічний інтерфейс користувача (GUI), реалізований на Qt.
   b. Клієнт повинен надавати інтерфейс для реєстрації, входу та відновлення пароля.
   c. Клієнт повинен автоматично отримувати та зберігати конфігураційний файл VPN після успішної авторизації.
   d. Клієнт повинен мати можливість ініціювати та розривати VPNз'єднання, взаємодіючи з локально встановленим процесом OpenVPN.
   e. Клієнт повинен відображати список груп, до яких належить користувач, та надавати інтерфейс для управління групами (якщо користувач є їх власником).
   f. Клієнт має реалізовувати ssh-клієнт/сервер для реалізації віддаленого доступу.

Потенційні напрямки розвитку

Мета: Розширення базового функціоналу для створення повноцінної платформи для спільної роботи та віддаленого адміністрування.

Список функцій для майбутньої реалізації:

1. Повноцінне динамічне керування групами доступу:
    a. Створення та керування групами користувачів через клієнтський інтерфейс.
    b. Реалізація на сервері динамічного керування мережевими правилами доступу між групами за допомогою nftables.
2. Віддалений доступ з відеотрансляцією:
    a. Захоплення екрана та кодування відео за допомогою бібліотек FFmpeg.
    b. Організація потокової передачі даних в реальному часі (спочатку P2P через VPN, у подальшому — через SFU, наприклад, Janus Gateway).
    c. Декодування та відображення відеопотоку в Qt-віджеті (QOpenGLWidget).
3. Розширене керування сесіями та клієнтом:
    a. Повна реалізація системи Access та Refresh токенів.
    b. Кросплатформена компіляція та запуск клієнта під ОС Windows.
4. Інфраструктура та розгортання:
    a. Перенесення серверної частини в Docker-контейнери для спрощення розгортання.

Кроки реалізації продукту

- Етап 1 (MVP): Ядро проєкту:
  - Реєстрація/авторизація/верифікація користувачів.
  - Автоматична генерація конфігураційних файлів OpenVPN та сертифікатів.
  - Клієнт, який може залогінитись, завантажити конфіг і вручну запустити його через встановлений OpenVPN клієнт.
  - Інтеграція клієнта з процесом OpenVPN (запуск/зупинка з GUI), керування групами та правилами nftables.
- Етап 2: Реалізація відеотрансляції.
- Етап 3: Реалізація віддаленого контролю.