Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка програмного комплексу приховування vpn активності за допомогою обфускації трафіку

Виконав: студент 4 курсу, гр.1КІТС-21б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем

Палюх М. Є. _____

Керівник: к.ф.-м.н., доцент каф. МБІС

Шиян А. А. _____

«____» _____ 2025 р.

Рецензент: к.т.н., доц., доцент каф. ОТ

Богомолов С. В. _____

«____» _____ 2025 р.

Допущено до захисту

Голова секції УБ кафедри МБІС

д.т.н., проф. Яремчук Ю.Є.

“____” _____2025р.

Вінниця ВНТУ – 2025

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю.Є.
" 20 " березня 2025 р.

З А В Д А Н Н Я
на бакалаврську кваліфікаційну роботу студенту
Палюху Матвію Євгенійовичу
(прізвище, ім`я, по-батькові)

1. Тема роботи Розробка програмного комплексу приховування vpn активності за допомогою обфускації трафіку
Керівник роботи к.ф.-м.н., доцент кафедри МБІС Шиян Анатолій Антонович
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 20 " березня 2025року № 97
2. Термін подання студентом роботи за тиждень до захисту
3. Вихідні дані до роботи Електронні джерела, офіційні онлайн репозиторії використаних протоколів, що були представлені у бакалаврській роботі, наукові статті.
4. Зміст текстової частини:
У першому розділі розглядаються теоретичні засади предметної області та розкриття проблеми, пов'язаної з темою бакалаврської роботи, у другому розділі наводяться пояснення алгоритмір роботи протоколів та у третьому розділі відбувається реалізація програмного комплексу та аналіз його роботи.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень) UML діаграма роботи системи при TLS сесії. UML діаграма роботи системи при звичайній взаємодії. Блок-смеха роботи системи. Інтерфейс клієнту проксування. Налаштування профілю проксування. Аналіз роботи звичайного TLS зв'язку.

Аналіз роботи протоколу OpenVPN. Аналіз роботи протоколу Wireguard. А
роботи розроблюваної програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Шиян А. А., к.ф.-м.н., доц. каф. МБІС		
Розділ II	Шиян А. А., к.ф.-м.н., доц. каф. МБІС		
Розділ III	Шиян А. А., к.ф.-м.н., доц. каф. МБІС		

7.Дата видачі завдання 20 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Визначення теми бакалаврської роботи	20.03.2025	23.03.2025	Виконано
2	Аналіз предметної області обраної теми	24.03.2025	13.04.2025	Виконано
3	Розробка алгоритму роботи	14.04.2025	27.04.2025	Виконано
4	Написання бакалаврської роботи	28.04.2025	25.05.2025	Виконано
5	Передзахист бакалаврської кваліфікаційної роботи	26.05.2025	27.05.2025	Виконано
6	Виконання правок та виправлень бакалаврської кваліфікаційної роботи	28.05.2025	02.06.2025	Виконано
7	Захист бакалаврської кваліфікаційної роботи	09.06.2025	10.06.2025	Виконано

Студент Палюх М. Є.
(підпис) (прізвище та ініціали)

Керівник роботи Шиян А. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 81 сторінок формату А4, на яких є 19 рисунків, 2 таблиці, список використаних джерел містить 22 найменування.

Метою роботи є розробка програмного комплексу, що буде забезпечувати маскування інтернет протоколів, які використовує користувач під час проведення діяльності в мережі інтернет, зокрема, протоколів що забезпечують послуги проксування та створення VPN-тунелів між сервером проксування та користувачем.

У роботі буде розглянуто основи роботи мережі інтернет, обміну даними у ній, принципи роботи технології проксування та створення VPN-тунелів. Буде описана проблема, яка існує з сьогоденною приватністю в мережі інтернет для користувача, та потреба використання технологій мережевої обфускації трафіку, та власне, принцип її роботи.

Буде описано складові програмного комплексу та схему його роботи, використані протоколи обфускації та технології, що забезпечують взаємодію системних складових кінцевих пристроїв з цими протоколами.

Насамкінець, буде проведено тестування ефективності створеного програмного комплексу, та проведено порівняння обфускованого трафіку до не замаскованого трафіку.

Ключові слова: інтернет, проксі, VPN-тунель, обфускація, мережевий трафік, v2ray, tls, proxy.

ABSTRACT

The bachelor's thesis consists of 81 pages of A4 format, on which there are 19 figures, 2 tables, the list of used sources contains 22 names.

The aim of the work is to develop of a software complex that should ensure the masking of Internet protocols used by the user when conducting activities on the Internet, in particular protocols that provide proxy services and the creation of VPN tunnels between the proxy server and the user.

The work will cover the basis of the Internet, data exchange in it, the principles of proxy technology and the creation of VPN tunnels. The problem that exists with the current privacy on the Internet for users, and the need to use network traffic obfuscation technologies, and, in fact, the principle of its operation, will be described.

The components of the software complex and the scheme of its operation will be described, obfuscation protocols and technologies that ensure the interaction of system components of end devices with these protocols will be used.

Finally, the effectiveness of the created software package will be tested, and the obfuscated traffic will be compared with the unobfuscated traffic.

Keywords: internet, proxy, VPN tunnel, obfuscation, network traffic, v2ray, tls, proxy.

ЗМІСТ

ВСТУП 7
1 ТЕОРЕТИЧНІ ЗАСАДИ ОБФУСКАЦІЇ ІНТЕРНЕТ ТРАФІКУ 9
1.1 Робота мережі інтернет та технологій проксі з VPN 9
1.2 Основи роботи VPN та проксі протоколів 15
1.3 Методи аналізу інтернет трафіку 19
1.4 Принцип роботи обфускації трафіку та огляд декількох технологій 22
1.5 Аналіз аналогів програмного комплексу з обфускації 25
1.6 Висновки до розділу 1 27
2 УДОСКОНАЛЕННЯ ЗАСОБУ ОБФУСКАЦІЇ ТРАФІКУ 28
2.1 Обґрунтування удосконалення існуючих способів обфускації трафіку 28
2.2 Опис середовища розробки, мови програмування 33
2.3 Проектування UML діаграм послідовностей та блок-схеми роботи програмного комплексу 35
2.4 Висновки до розділу 2 41
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ 42
3.1 Налаштування серверної частини програмного комплексу 42
3.2 Налаштування клієнтської частини програмного комплексу 46
3.3 Аналіз та порівняння звичайного та VPN трафіку з обфускованим трафіком 50
3.4 Висновки до розділу 3 57
ВИСНОВОК 59
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 61
ДОДАТКИ 66
Додаток А. Технічне завдання 67
Додаток Б. Лістинг коду файлу конфігурації V2Ray серверної частини комплексу 71
Додаток В. Ілюстративний матеріал 74
Додаток Г. Протокол перевірки на антиплагіат 81

ВСТУП

Актуальність. Під час взаємодії користувача з мережею інтернет, весь його мережевий трафік, проходить через сервери його регіонального провайдера інтернет послуг, ISP. Провайдер, при бажанні, здатен проводити аналіз трафіку який надходить від певного користувача, за допомогою аналізаторів трафіку, так званих – сніферів. Сніфери можуть проводити поверхневий огляд пакетів трафіку або глибинний, визначаючи адресу призначення трафіку та його вміст його пакетів.

Якщо надавач послуг інтернет отримав наказ від державних органів на фіксування IP-адрес користувачів цього ISP, при їхньому підключені до певного списку IP-адрес, або при використанні ними технологій проксування, зокрема з утворенням VPN-тунелів, то користувачу, щоб уникнути подальшої небезпеки через це, та можливих перевірок, варто почати використовувати технології маскування такого трафіку, під "звичайний" трафік передачі даних в інтернеті, що називається обфускацією трафіку. Стосуватись така небезпека може і користувачів на окупованій території України.

Мета роботи. Метою цієї роботи є створення програмного комплексу, що буде забезпечувати обфускацію трафіку, за утворення якого, користувач піддається небезпеці з боку репресивного державного режиму, що діє на території перебування користувача. Відповідно, було поставлено набір завдання для досягнення цієї мети:

– проаналізувати ознаки робіт VPN технологій у мережевому трафіку;

– розробити набір технологій, що забезпечить надання VPN послуг користувачеві, та не буде відображати відомих ознак VPN технологій під час своєї роботи;

– побудувати програмний комплекс, що складається з серверної і клієнтської частини, та забезпечить обфускацію VPN трафіку.

Об’єктом дослідження є створення програмного комплексу, який забезпечить обфускацію VPN трафіку.

Та **предметом дослідження** є інкапсуляція пакетів VPN трафіку в не підозрілий транспорт, позбавлення ознак вмісту пакетів VPN у ньому та зміна статистичних характеристик трафіку.

Новизна роботи. Полягає створенні та налаштуванні такого програмного комплексу, що для надання VPN послуг, використовує поєднання технологій VLess та XTLS-Vision, та який на відміну від своїх аналогів, VPN клієнтів, забезпечить приховання всіх ознак використання користувачем VPN технології.

Практична цінність роботи. У роботі було використано передові, на сьогоднішній день, технології обфускації, та наведено інструкції, для їх імплементації на системі користувача. Для доказу ефективності, були приведені відповідні наукові дослідження та деякі відомості, що до їх ефективності їх роботи від користувачів, що випробовують їх на цензурі їхніх ISP.

1 ТЕОРЕТИЧНІ ЗАСАДИ ОБФУСКАЦІЇ ІНТЕРНЕТ ТРАФІКУ

У цьому розділі розглянуто ключові поняття та технології, що стосуються інтернет-трафіку, механізмів його контролю, методів аналізу та сучасних підходів до обфускації. Теоретична база, що наведена в цьому розділі, слугуватиме основою для практичної реалізації програмного комплексу з обходу інтернет-цензури та оцінки його ефективності у другому розділі. Алгоритми роботи VPN протоколів та способи аналізу мережевого трафіку, наведені у цьому розділі, будуть розглядатись та використовуватись наочно у третьому розділі.

1.1 Робота мережі інтернет та технологій проксі з VPN

Використавши цитування з одного відкритого джерела, інтернетом є всесвітня система сполучених комп'ютерних мереж, що заснована на комплекті інтернет протоколів. Інтернет також називають мережею мереж, бо він складається з мільйонів локальних і глобальних приватних, публічних, академічних, ділових і урядових мереж, пов'язаних між собою з використанням різноманітних дротових, оптичних і бездротових технологій [1].

Відповідно, мережа інтернет дозволяє користувачам проводити обмін цифровою інформацією всюди, де вона пролягає. Для того щоб, поки що особі, отримати можливість використання інтернет мережі, їй необхідно стати клієнтом її місцевого надавача інтернет послуг, ISP, після чого вона стає користувачем інтернет мережі. ISP в свою чергу, забезпечує мережеву взаємодію клієнта з іншою, зовнішньою, мережею шляхом передачі її трафіку до провайдерів інтернету, що обслуговують таких ISP та інших серверів, що надають інформаційні інтернет послуги всім бажаючим, тим самим утворюючи ієрархію мережевої адресації по всій землі.

Надавачі інтернет послуг кінцевого користувача, зазвичай відносяться до найнижчої ланки розподілення інтернет адресації, так звана Tier 3, далі по ієрархії, стоять регіональні або національні провайдери інтернет послуг, що обслуговують Tier 3 ланки, але не кінцевих користувачів, так звані Tier 2. Та вже представникам

Tier 2 ланки, послуги зв’язку надають глобальні провайдери інтернету, які вже безпосередньо "спілкуються" між собою утворюючи глобальну мережу інтернет, та власне забезпечуючи глобальну адресацію для всіх представників Tier 2 ланок, у країнах де вони є, відповідно стаючи Tier 1 ланкою [23].

На ланках Tier 3 та Tier 2, можлива реалізація регулювання та контролю роботи інтернет мережі для певної території, здебільшого країни, де вони працюють. Підпорядковуються вони державним регулюючим органам, про небезпеку такої взаємодії, яка створюється для користувача такого провайдера, буде мовитись далі.

Робота інтернет мережі складається з рівнів, оскільки необхідно забезпечити фізичне середовище передачі цифрової інформації, сумісність різних типів апаратного устаткування, що бере участь у передачі інформації, локальну та глобальну адресацію та форму даних, що власне отримує користувач. Ці рівні мають назву – мережева модель OSI та складається вона з семи рівнів, всі з яких існують там де пролягає мережа інтернет. Найнижчі два рівні визначають які у пристрою повинні бути модулі, фізичні порти, для під’єднання фізичного середовища передачі інформації, які характеристики повинні мати фізичні середовища передачі інформації, та алгоритми кодування сигналу, що до них надходить, а також, забезпечення можливості взаємодії пристроїв з різними мережевими інтерфейсами. Далі, третій рівень, забезпечує можливість локальної та глобальної адресації при передачі трафіку, шляхом призначення Internet Protocol адреси необхідних пристроям. Здебільшого, користувач інтернету не має своєї власної глобальної адреси, але має наявне підключення до шлюзового маршрутизатора, який вже її має. Шлюзовий маршрутизатор, може мати декілька таких локальних користувачів, всі з яких користуються однією IP-адресою, та саме завдяки порту четвертого рівня розрізняються у глобальній мережі. На цьому моменті необхідно зазначити, що саме на фактичні адресі проживання користувача інтернет знаходиться його особистий шлюзовий маршрутизатор, ці дані має його місцевий надавач послуг, та ці дані імовірно є доступними до місцевих державних представників. Четвертий рівень забезпечує визначення послуг, що їх потребує

користувач. Називається це послугами транспорту та реалізуються двома основними протоколами TCP та UDP, різниця їх полягає у швидкості обробки, коли пакет з таким транспортом надходить на вузол, та захисті даних, що їх несе пакет. А також транспортний рівень забезпечує розмежування різних послуг що може надавати пристрій з певною IP-адресою, таким чином, одна IP-адреса може надавати десятки тисяч різних послуг, або розрізняти тисячі різних користувачів інтернету, все це реалізується номером порту. Потреби у розгляді локальної адресації, для подальшого виконання роботи немає, проте, необхідно зазначити, що мережевий пристрій користувача, при підтримці третього та четвертого рівня OSI, має можливість звернення до своєї локальної адреси, так званий локальний хост, що складається з комбінації IP-адреси та номеру порту, так званий сокет. Трафік, що адресований на адресу локального хосту, не покидає його, а надходить, або проходить через ще одну програмну ланку всередині пристрою, створюючи зручний спосіб контролю трафіку, що вже буде надходити за межі пристрою, так званий системний проксі. Та п’ятий та вищі рівні, є місцем реалізації роботи різних протоколів, що працюють з безпосередньою інформацією користувача, подаючи її у різних формах та надаючи різноманітний функціонал для обміну інформації, наприклад покращення швидкості обміну, використання додаткового шифрування, пришвидшення роботи передачі пакетів, більш ефективне використання пропускної здатності каналу передачі даних, та багато іншого, так званий прикладний рівень.

Пакет інформації утворюється шляхом інкапсуляції заголовків з одного рівня в інші, рисунок 1.1. Де заголовки другого рівня Network access layer, містять в собі заголовки третього рівня Network layer, які також містять в собі заголовки четвертого рівня Transport layer, які вже містять дані Payload, користувача, відповідно п’ятий та вищі рівні моделі OSI.

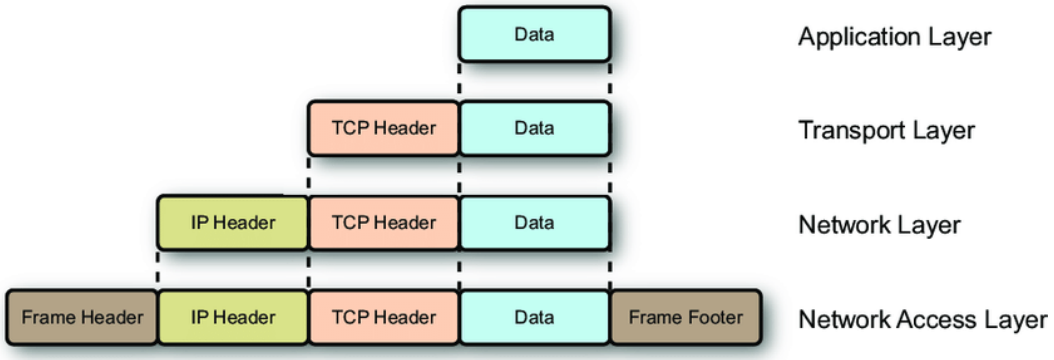


Рисунок 1.1 – Багаторівнева структура інтернет кадру [3]

Вся інформація що містися від першого до четвертого рівня включно, є відкритою, і зокрема для провайдера. Інформацію, що переносе в собі заголовок четвертого рівня, може бути повністю зашифрована, та навіть не мати явних ознак використання певного протоколу прикладного рівня. При використанні певних протоколів у прикладному рівні, у заголовках транспортного рівня зазвичай зазначаються відповідні номери портів, що прийнято використовувати при тому чи іншому протоколі прикладного рівня. Ця залежність не є обов’язковою, проте зазвичай використовується для коректної роботи мережевої взаємодії. Ця залежність є відкритою, та відповідно, дозволяє ідентифікувати протокол що працює на прикладному рівні, тим не менш, цим можна знехтувати. Це необхідно буде врахувати при виконанні роботи.

Таким чином, ISP користувача та Tier 2 провайдер цього ISP, завжди "бачать" до якої IP-адреси звертається користувач, та найбільш імовірно – яку послугу бажає використати. Ця взаємодія зображена на рисунку 1.2, де WWW позначає представника служби інтернету, який відповідно, має свою IP-адресу.

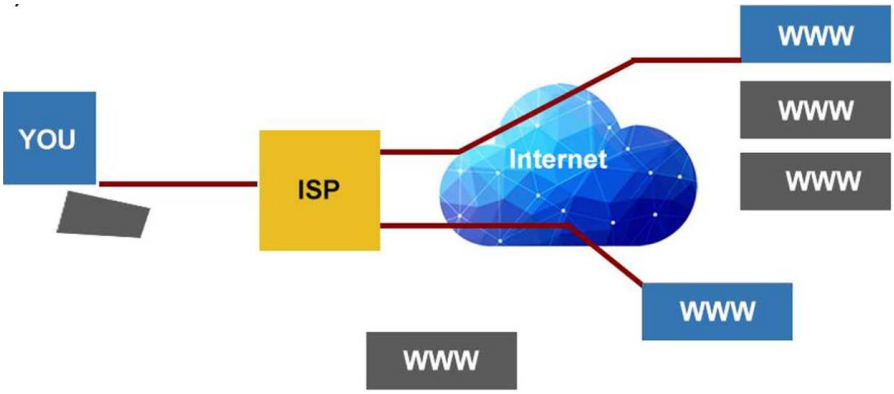


Рисунок 1.2 – Стандартна робота мережевої взаємодії [2]

При цьому, якщо користувач не може під’єднатись до певних представників через блокування доступу до них на рівнях Tier 3 та Tier 2, або, йому необхідно не допустити, щоб ці провайдери не дізнались до яких саме адрес він звертається та якими послугами користується, він може почати використовувати технологію проксування свого трафіку. Таким чином, провайдери Tier 3 та Tier 2 рівня, будуть бачити, що користувач весь час звертається до IP-адреси та користується такими послугами, які по-перше, не підлягають блокуванню, а по-друге, є допустимими до виявлення його провайдерами.

Проксування існує декількох видів, робота буде виконуватись з використанням прямого проксування. Пряме проксування починається з системного налаштування на стороні клієнта, коли програма, яка вимагає мережевої взаємодії створює інтернет трафік, він не покидає пристрій одразу, а підлягає змінам на ланці нового мережевого інтерфейсу, або на ланці локального хосту, про який мовилось раніше. Далі, незалежно від протоколу що використала програма, її трафік змінюється на перелік інструкцій, що необхідно буде виконати проксі серверу, відповідно, змінюється і оригінальний прикладний протокол який використала програма, на протокол, яким власне буде відбуватися взаємодія з проксі сервером. Далі трафік покидає пристрій та надходить тільки на IP-адресу проксі сервера, де отриманий трафік проходить ідентифікацію, на дозвіл до обробки. Після успішної ідентифікації, проксі згідно інструкцій виконує запит до кінцевого ресурсу, який в свою чергу, нічого не знатиме про користувача.

Далі, розглянемо технологію VPN – virtual privat network. Цей концепт був створений для забезпечення надбезпечної передачі даних в інтернеті, та прийшла на заміну фізичним виділеним лініям інтернет зв’язку. Оскільки прокладання фізичних ліній є досить дорогим рішення за для безпеки, а для великих відстаней, наприклад міжконтинентальних – неможливим рішенням, було винайдено стек технологій під назвою virtual privat network [24]. VPN окрім шифрування, повинен забезпечувати надійну ідентифікацію двох сторін контакту, захист проти змін у ході надсилання та інші властивості. Первинно був створений, щоб працівники одного об’єднання мали можливість проводити повністю захищену взаємодію на далеких відстанях, для віддалених працівників. Здатен працювати в тунельному режимі, створюючи VPN "тунель" від шлюзу маршрутизації до IP-адреси призначення, або у режимі від пристрою користувача до IP-адреси призначення. Оскільки VPN технології створено для корпоративного використання, для того щоб користувачі під’єднувались до певних серверів компанії, що підтримують технології VPN, виникає проблема якщо користувач бажає використати VPN технологію при зверненні до серверів, які вже не є власністю його корпорації, тобто вся інша мережа інтернет. І тут стає доцільним поєднання технологій проксування та VPN, проводячи захищене підключення за допомогою VPN протоколів до проксі, а він вже в свою чергу, буде виконувати звернення до кінцевих, не корпоративних, ресурсів, та сторонній спостерігач на проміжку від проксі до кінцевого вузла не буде знати істинно від кого та до кого надходе трафік [25]. Саме таким чином, утворені комерційні клієнти, що встановлюються на пристрій користувача та надають послуги проксування з VPN технологіями, та їх прийнято називати VPN клієнти, сервіси, їх робота зображена на рисунку 1.3.

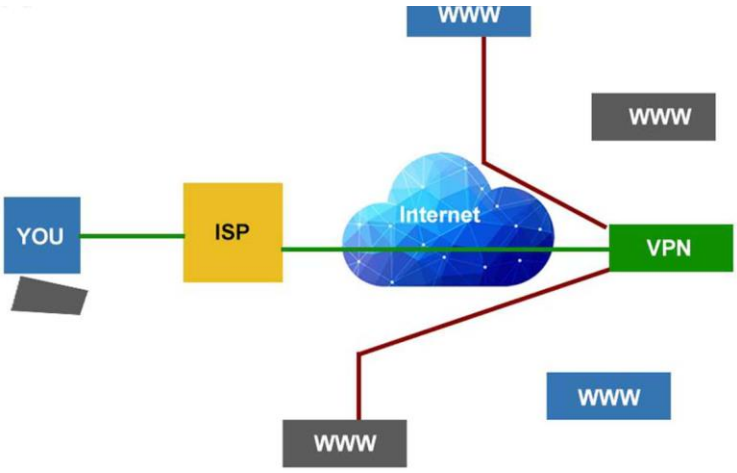


Рисунок 1.3 – Мережева взаємодія з використання технології VPN [2]

Найпопулярніші у виборі використання VPN протоколи таких клієнтів працюють з безпекою протоколу TLS. На сьогоднішній день, він використовується великою кількістю прикладних протоколів, що надають основний спектр послуг мережі інтернет, наприклад HTTPS – перегляд вебсайтів, FTPS – передавання файлів, SMTPS – відправка електронної пошти, IMAPS – отримання електронної пошти та інші [45].

1.2 Основи роботи VPN та проксі протоколів

У цьому підрозділі буде розглянуто, як працюють VPN протоколи, власне які використовують готові VPN клієнти, та як цю роботу помітно у мережі, зокрема ISP користувача. VPN технологія поміщає пакет, який утворила програма користувача в оболонку прикладного рівня, після чого відправляє його IP-адресі певного сервера, що власне працює по цій VPN технології, та налаштований на отримання трафіку від цього користувача, після чого розпаковує VPN пакет, отримуючи "сирий" пакет програми користувача, та згідно своїх налаштувань, пересилає його до локальної мережі цього сервера – у разі корпоративного використання, або у відкритий інтернет – варіант роботи проксі. Якщо такий VPN сервер має відповідне налаштування, розпакувавши "сирий" пакет користувача, перед його відправкою далі IP-адресі призначення кінцевого ресурсу, сервер проводить зміну його

заголовків, забезпечуючи анонімність користувачу. Таким чином, VPN протоколи з початку своєї появи, забезпечували послуги дуже схожі на послуги проксування.

Отже, спершу розглянемо які існують основні протоколи проксування, власне з якими можуть працювати VPN протоколи та сервери.

HTTP проксі може проксувати тільки запити HTTP протоколу, відповідно сьомий рівень OSI, при чому коли програма користувача утворює HTTP пакет, він, на етапі підготовки до перенаправлення до проксі сервера, підлягає зміні у деяких його заголовках, та коли проксі сервер отримує цей запит, він виконує його від свого "імені". По замовчуванню не підлягає шифруванню, проте можливо встановити TLS сесію з проксі сервером, щоб забезпечити безпеку передачі. Працюючи за допомогою протоколу HTTP, не підтримує проксування інших прикладних протоколів, наприклад пошти, файлової передачі тощо. Відповідно не працює з транспортом UDP, допускає роботу в режимі прозорого проксі, transparent proxy, при чому звернення до кінцевого ресурсу відбувається з IP-адреси користувача, не проксі. Зручний, оскільки роботу з ним підтримують більша кількість програм та процесів користувача, ніж з SOCKS проксі.

SOCKS проксі, Secure Socket proxy, працює на сеансному рівні моделі OSI, дозволяючи проксування усіх прикладних протоколів. При утворенні пакету програмою користувача, пакет не підлягає змінам, одразу надходячи до проксі сервера, який в свою чергу, не має можливості працювати з у прозорому режимі. Проксі сервер змінює пакет, так наче він надходе від нього. SOCKS підтримує роботу з UDP транспортом та не має вбудованого шифрування. Забезпечення безпеки можливе з додаванням TLS шифрування або SSH тунелю [26]. Здатен працювати швидше ніж HTTP проксі, проте залишається проблема меншої популяризації в підтримці його роботи серед системних програм, наприклад, коли необхідно щоб програма перенаправляла свої пакети спочатку до адреси локального хосту протоколом SOCKS, а вже він буде надавати їх мережевому інтерфейсу, про це буде мовитись далі в роботі.

Тепер розглянемо деякі найвідоміші VPN протоколи та за якими ознаками їх можна виявити у трафіку.

OpenVPN працює наступним чином, програма користувача створює свій пакет, який має всі заголовки від найвищого протоколу до мережевого, IP-адреси [27]. Після чого пакет передається до віртуального мережевого інтерфейсу, де він може пройти стиснення, проходить шифрування та інші міри захисту від OpenSSL бібліотеки TLS протоколу, отримуючи нову IP-адресу та інші заголовки, таким чином "сирий" пакет програми інкапсулюється у TLS тунель на прикладному рівні, отримавши цей пакет, сервер проводить зворотні дії, та його подальше перенаправлення, рисунок 1.4.

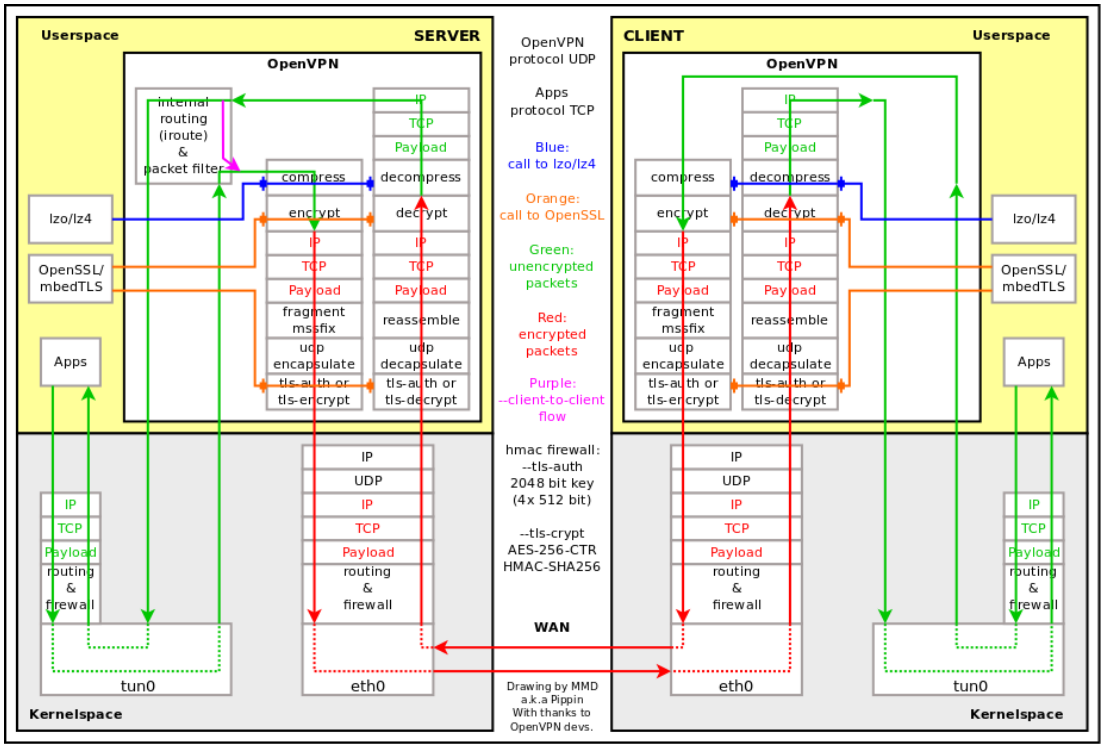


Рисунок 1.4 – Алгоритм роботи протоколу OpenVPN [4]

При цьому, пакети які покидають фізичний інтерфейс користувача, можуть бути ідентифіковані як пакети OpenVPN протоколу за наступними ознаками. Кожен заголовок OpenVPN, рисунок 1.5, починається з коду операції, який визначає тип повідомлення поточного пакета. Поле коду операції може приймати понад 10 визначених значень, що відповідають типам повідомлень, що передаються на різних етапах зв'язку. Типовий сеанс OpenVPN починається з надсилання клієнтом повідомлення Client Reset. Потім сервер відповідає повідомленням Server Reset, після чого відбувається встановлення зв'язку TLS. Пакети OpenVPN, що містять

шифротексти TLS, мають тип повідомлення P-Control. Оскільки OpenVPN може працювати через UDP, але повинен забезпечувати надійний канал для TLS, кожен пакет P-Control явно підтверджується пакетами P-ACK. Нарешті, фактичні корисні навантаження передаються як пакети P-Data.

Рисунок 1.5 – Структура пакетів протоколу OpenVPN [5]

Далі розглянемо MS SSTP, Secure Socket Tunneling Protocol, MS у назві позначає розробника, та протоколи які він використовує є схожими з попереднім протоколом. Пакети які створює програма користувача, переходячи на віртуальний інтерфейс, поміщаються у PPP протокол, далі, PPP протокол поміщається у SSTP заголовок, а той вже інкапсулюється у HTTPS заголовок, таким чином, основне шифрування, як і в попередньому варіанті, забезпечується TLS протоколом, проте у даному випадку, MS SSTP працює тільки за TCP транспортом. Канальний протокол PPP забезпечує додаткову послугу аутентифікації з інтерфейсом призначення, та послуги керування сесією, PPP не використовує MAC адресації, оскільки цей протокол був розроблений для підключення точка-точка, саме так як і працює VPN тунель, що не потребує різної адресації за допомогою MAC адрес – таку функцію забезпечує інший канальний протокол Ethernet, що власне у сценарії роботи MS SSTP не становить потреби. Проте, адресація PPP все ж таки використовується – широкомовна. З попереднього опису HTTP проксі, можна припустити, що MS SSTP теж працюючи через HTTP інкапсуляцію, не можу переправляти інші протоколи, окрім HTTP, проте це не так – в цьому і полягає відмінність VPN технології від інших технологій. Проте виявити роботу протоколу можна, якщо виконати HTTP запит до сервера до якого звертається користувач, методом SSTP_DUPLEX_POST та з URI адресою /sra_{BA195980-CD49-458b-

9E23-C84EE0ADCD75}/, в результаті чого, сервер відповість підтвердженням наявності MS SSTP [6].

Наступним розглянемо Wireguard, цей протокол є найновішим з представлених у цьому підрозділі, він не використовує TLS протоколу, та використовує сталий набір протоколів для проведення шифрування, обміну ключами симетричного шифрування, хешування та інші [28]. Працює тільки транспортом UDP, тим самим забезпечуючи кращу швидкість. Створений пакет програми користувача інкапсулюється у заголовок Wireguard, який і забезпечує функціонування протоколів його єдиного стеку, далі заголовок Wireguard отримує IP-адресу та UDP заголовок, після чого проводиться відправка. Власне виявити його можна за декількома ознаками, мінімальний розмір корисного навантаження UDP повинна скласти 32 байти – 16 початкова частина та 16 фінальна частина Wireguard заголовку, маючи 0 біт корисного навантаження у собі, так званий keepalive пакет. Далі, перевірка чи корисне навантаження має 2-4 байти зарезервовані, тобто дорівнюють нулю, Також, якщо пакет має "ініціалізацію сеансу" – ID 0x01, відбуваються дві речі. По-перше, випадковий індекс відправника витягується з пакета. По-друге, цей індекс порівнюється з попереднім індексом, що зберігається в завантаженому потоці [7].

Протокол Ipsec, найстаріший з усіх представлених, дозволяє поєднувати різні протоколи шифрування, аутентифікації, які використовуються здебільшого ним, тому і виявлення майже завжди є точним, працює тільки з UDP транспортом [43].

Тому користувач піддається небезпеці з боку представників влади території на якій він знаходиться, якщо його ISP отримає наказ про фіксування IP-адрес користувачів та їх фізичних адрес у разі утворення ними VPN трафіку. Відповідно було розглянуто за якими ознаками можна ідентифікувати VPN та проксі протоколи.

1.3 Методи аналізу інтернет трафіку

Дослідження роботи програмного комплексу з обфускації трафіку буде відбуватися аналізом трафіку, що покидає пристрій користувача. Проводитись це

дослідження буде у третьому розділі. Для аналізу буде використовуватись програма Wireshark, оскільки вона є досить зручним інструментом для аналізу трафіку, що має багато корисних функцій. Wireshark дозволяє захопити трафік що надходить з певного програмного чи апаратного мережевого інтерфейсу пристрою користувача, провести його аналіз та при бажанні зберегти трафік у .pcap, .txt та інших форматах [44].

Для аналізу буде використовуватись ручний метод, при якому буде проводитись розбір складових пакету, його вмісту, та статистичний аналіз трафіку, про що буде описано далі.

При запуску Wireshark користувачу надається вибір, з якого мережевого інтерфейсу він бажає почати фіксування трафіку. Це вікно надає можливість, вже на основі певного джерела трафіку, додати фільтр або інше налаштування при утворенні нового інтерфейсу фіксування трафіку. Та для зручнішого виконання роботи, було створено новий інтерфейс фіксування трафіку з додатком налаштуванням, що додає можливість відслідкування походження пакету, який надійшов на мережевий інтерфейс. А саме, процесу від якого він був створений, це дозволить ідентифікувати програму, що утворила пакет, і власне трафік, що дозволяє, при бажанні, ввести фільтр на певний Process ID, PID, та отримати весь потік, stream, пакетів що утворив певний процес, від початку до кінця. Додаткове налаштування має такий вигляд: "--p=Microsoft-Windows-NDIS-PacketCapture". Обравши цей, або інший мережевий інтерфейс, програма дозволяє застосувати до вже захопленого або в майбутньому захопленого трафіку певний фільтр, фільтрацію можна застосувати до будь-яких атрибутів, що мітять пакети трафіку. Wireshark, надає вміст пакету у шістнадцятковій системі числення, та проводить автоматичне резолювання та детекцію протоколів, що містяться у пакеті, що дозволяє користувачу зручно, по замовчуванню англійською мовою, ознайомитися з вмістом пакету, та провести його ручний аналіз. Фільтрацію можна застосувати, наприклад, до певної IP-адреси джерела походження пакету, протоколу, розміру пакету, до певного вмісту у пакеті, наприклад вже згаданий PID процесу походження та інші фільтри. Є можливість об’єднання фільтрів. Також, зручною властивістю Wireshark,

є можливість отримати об’єднання корисних навантажень певного потоку пакетів, що утворюються певним протоколом, наприклад TCP, або TLS, у вигляді звичайного тексту, де у кожного корисного навантаження присутнє своє посилання на пакет, звідки воно походить. Wireshark здатний розпізнавати та вести слідкування за пакетами, що утворені та належать одному потоку, stream [29].

Перейдемо до можливостей статистичного аналізу трафіку в Wireshark. Для статистичного аналізу необхідно обрати трафік, над яким воно буде проводиться, для цього, вона створити .pcap файл, який буде повністю складатись з пакетів, що необхідно буде дослідити, або ти кожному з статистичних аналізів, необхідно буде застосовувати фільтри трафіку. Наприклад певний PID, який міститься у пакетах, певна IP-адреса, певний потік визначеного протоколу, власне певний діалог між IP-адресами та інші.

Перш за все, статистичний аналіз демонструє загальну інформацію про відфільтрований трафік, кількість пакетів, обсяг всіх пакетів у байтах, час запису та інші, більшість з яких містяться на головному інтерфейсі. Наступним можна провести дослідження ієрархії протоколів серед відфільтрованого трафіку, тобто, відображення кількості пакетів що належать певному протоколу, та оскільки прикладні протоколи інкапсулюються у транспортні протоколи та інші, нижчі рівні моделі OSI, це все зазначається ієрархічною структурою при відображенні. Наступним статистичним аналізом є відображення розмірів пакетів у відфільтрованому трафіку. Що у відсотковому значені показує кількість пакетів у трафіку в своїх діапазонах розмірів, що може стати корисним для знаходження аномально великих чи навпаки замалих пакетів та інших характеристик. І на останок, I/O графік, показує графік залежності часу від кількості пакетів, що надійшли на інтерфейс. Вікно I/O графіку дозволяє вибрати які саме пакети необхідно піддати залежності від часу, та відповідно утворити графік. Зробити це можна шляхом тої самої фільтрації. Цей графік зможе показати, ознаки ініціації зв’язку певного протоколу та тип трафіку, що проходить через інтерфейс. Всього існує три види трафіку, відео трафік, голосовий та інформаційний трафік. VOIP голосовий трафік у цій роботі розглядатись не буде. Основна різниця на I/O графіку

між відео та data трафіком є їхня періодичність, де у data трафіку, відбувається разовий обмін даними, та потім хоч навіть 5 хвилин "тиші" після чого знову може статись обмін даними, прикладом є перегляд HTML сторінок, а у відео трафіку відбувається обмін набору досить містких пакетів з однаковою періодичністю, та майже відсутність режиму тиші між передачею цих наборів. Таким чином, були наведені основні методи аналізу трафіку та відповідно оцінки ефективності результату роботи, які будуть застосовані в третьому розділі [42].

1.4 Принцип роботи обфускації трафіку та огляд декількох технологій

Отже, якщо користувачу необхідно приховати свою мережеву активність, IP-адреси до яких він звертається, протоколи які він використовує, щоб убезпечитись він можливого покарання з боку місцевої влади, він може почати використовувати технології обфускації свого трафіку. Обфускація походить від англійського слова "obfuscation" – заплутування, та має два основних підходи до реалізації – стеганографічні та поліморфні. Мета стеганографічного підходу це зробити обфускований трафік схожим на дозволений трафік, а мета поліморфізму це зробити трафік обходу не схожим на заборонений трафік. Для виконання стеганографічного підходу використовуються імітаційний та тунельний метод, останній показує себе краще у протидії виявленню, оскільки імітація протоколу є принципово недосконалою [8]. Проте, при використанні тунелювання все одно, необхідно забезпечити ідеальне узгодження ознак роботи протоколу з проведенням обфускації та без неї. Прикладом може стати виявлення та блокування пакетів, що були обфусковані тунельним методом в межах роботи мережі Tor на територіях Китаю та Ефіопії у 2012 році, імовірно йдеться за протокол Meek, у цій роботі не будуть розглядатися протоколи обфускації, що використовує тільки Tor [8]. На заміну цього підходу обирають поліморфний, поширений спосіб досягнення якого – повне шифрування корисного навантаження трафіку, без будь-якої відкритої інформації з заголовків або їх певної структури, на що і спираються засоби цензури при виявленні обфускації, недоліки, які при цьому залишаються будуть розглянуті далі. Саме такий метод буде використовуватись у виконанні роботи. Також,

необхідно розглянути проблему активного зондування, виконується воно коли, цензор, маючи підозру щодо активності користувача, від свого обличчя надсилає спеціально сформований пакет на IP-адреси, до яких звертається цей користувач, рисунок 1.6.

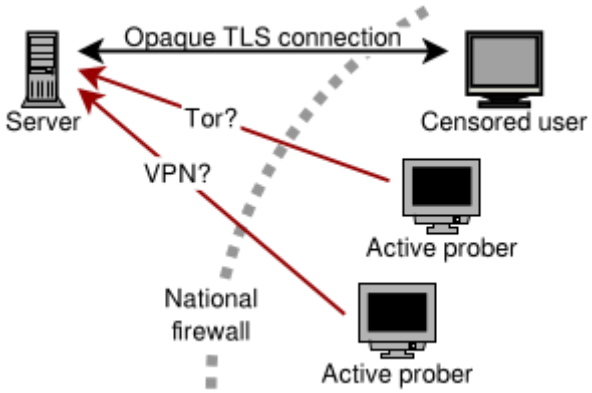


Рисунок 1.6 – Візуальне зображення активного зондування [9]

Отримавши такий пакет, чи набір пакетів, проксі сервер користувача імовірно надасть відповідь, яка імовірно буде містити свідчення про те, що це дійсно проксі сервер. Цей сценарій вже наводився у підрозділі 1.2, та його необхідно врахувати під час розробки програмного комплексу.

Програмний комплекс з обфускації може бути реалізований комбінацією різних технологій або монолітно. Почнемо з найчастішого у використанні протоколу обфускації – Shadowsocks. Цей протокол є реалізацією зашифрованого проксі протоколу SOCKS, зокрема його актуальної версії SOCKS5, працює він наступний чином, програма користувача формує SOCKS запит, та відправляє його до системного проксі або віртуального мережевого інтерфейсу, про це буде йтися далі у роботі, після, на цій системній ланці SOCKS запит шифрується за обраним алгоритмом симетричного шифрування, наприклад AES, ChaCha20 та інші, інкапсулюється у TCP чи UDP заголовок, та відправляється на проксі сервер [30]. Проксі сервер аутентифікує користувача за ключем, яким було зашифровано SOCKS запит. Оскільки використовується симетричне шифрування, проксі сервер необхідно попередньо налаштувати на певний протокол шифрування, та власне задати ключ шифрування, який буде співпадати з ключем користувача.

Існує можливість використання SSH тунелю як транспорту для проксі протоколів, як ще один спосіб обфускації. Захищений SSH тунель встановлюється між користувачем та проксі шляхом асиметричного шифрування, та потім без додаткового шифрування даних користувача або паддингу, відбувається перенаправлення SOCKS або HTTP пакетів програм користувача до цього тунелю [31].

На сьогоднішній день, найперспективнішим проектом з обфускації є V2Ray. Цей проект є фреймворком, який дозволяє, після його завантаження в систему, використовувати та поєднувати різні технології обфускації, таким чином створюючи велику кількість варіацій обходу цензури, саме він і буде використаний для виконання роботи. V2Ray містить як і вже оголошені способи обфускації так і власні, ефективні розробки. На вибір комбінації технологій, основним чином, вплинуло їх дослідження та оцінка їх ефективності з відкритих джерел, оскільки, через вже описану раніше у підрозділі, специфіку проблеми з російським цензором, існує можливість дізнатися який трафік він блокує, але немає можливості дізнатись який трафік він виявляє, саме цим і може скористатися цензор при пошуку та затримані "порушників". Тому далі буде приведено порівняння деяких технологій з фреймворку з оголошенням фінальної комбінації.

VMess є першою розробкою проекту V2Ray, цей протокол забезпечує проксування трафіку та його обфускацію. Працює він наступним чином програма користувача утворює запит за SOCKS протоколом та надсилає його до внутрішньої системної ланки, наприклад локальний хост, після, до нього додаються значення UUID, alterld та поточний час, значення UUID та alterld повинні сходитись у користувача та проксі сервері, оскільки далі відбувається шифрування утвореного пакету за допомогою AES або ChaCha20, використовуючи значення UUID та поточного часу при генерації, власне таким чином забезпечується аутентифікація користувача [32].

1.5 Аналіз аналогів програмного комплексу з обфускації

У цьому підрозділі буде проведений аналіз та порівняння аналогів з програмного комплексу, що надає послуги VPN з обфускації трафіку. Будуть розглянуті 3 варіанти, деякі з яких є готовими клієнтами. Вкінці буде побудована порівняльна табличка цих програм зі власною розробкою.

Mullvad VPN – це шведська компанія, що спеціалізується на конфіденційності та протидіє масовому стеженню та цензурі, має свій комерційний VPN-сервіс, який надає високий рівень анонімності, підтримує протокол WireGuard, як основний, та OpenVPN. WireGuard забезпечує швидкий і безпечний тунель, що було описано у першому розділі, проте не має вбудованої обфускації, тому для обходу цензури Mullvad використовує такі додаткові шари, як Shadowsocks та мостовий проксі. Mullvad не вимагає облікових даних, а лише анонімний ID, сервіс не збирає логи. Обфускація можлива, але не активна по замовчуванню. Захист від цензорів базовий, через просту структуру WireGuard, трафік виявляється без глибокого аналізу метаданих [33].

Outline VPN – це безкоштовний інструмент з відкритим кодом , який розгортає сервери Shadowsocks на кількох постачальниках хмарних послуг. Програмний пакет також включає клієнтське програмне забезпечення для кількох платформ. Outline був розроблений Jigsaw, технологічним інкубатором, створеним Google. Сервер Outline підтримує самостійне розміщення, а також хмарних постачальників послуг, включаючи DigitalOcean, Rackspace, Google Cloud Platform та Amazon EC2. Інсталяція передбачає виконання команди в інтерфейсі командного рядка або, у випадку інсталяції в DigitalOcean або Google Cloud, в графічному інтерфейсі користувача [17]. Outline VPN не є готовим VPN клієнтом, як попередній, а радше гібридом між повністю ручним налаштуванням VPN та клієнтами, де від користувача потребується "всього один натиск кнопки". Інструмент є простим у використанні, побудований на базі Shadowsocks. Призначений для журналістів і активістів, він має зручну адмін-панель для запуску сервера на VPS. Використовує AEAD-шифрування, наприклад ChaCha20-Poly1305, яке забезпечує конфіденційність трафіку. Outline VPN не має сильної стратегії

обфускації протоколу, наприклад за допомогою TLS або інших прикриттів, тому він є вразливим до DPI [34].

OpenConnect – програма з відкритим кодом що реалізує технологію віртуальної приватної мережі (VPN) для встановлення безпечних point-to-point з’єднань. Була розпочата як клієнт для VPN мережі Cisco AnyConnect, але з 2013-го включає в себе сервер і надає повне рішення зі створення VPN мереж. OpenConnect використовує SSL протокол AnyConnect – протокол VPN що користується TLS та DTLS для захисту трафіку всередині приватної мережі [18]. Для використання OpenConnect, потрібно мати власну орендовану VPS адресу зі встановленим Cisco сервером AnyConnect, ocserv, та вже з клієнту OpenConnect, проводити підключення до нього. Передає трафік через TLS/DTLS, UDP для більш швидкої передачі, схожий на HTTPS-тунель. Може виглядати як звичайне HTTPS-з'єднання, що забезпечує базову обфускацію. Проте, не має підтримки гнучкого обфускування, як VLess або Shadowsocks з uTLS, і може бути виявлений DPI-системами завдяки фіксованим шаблонам трафіку [35].

Таблиця 2.1 – Порівняння існуючих аналогів програмного комплексу з обфускації трафіку

Характеристика	Mullvad VPN	Outline VPN	OpenConnect	Власна розробка
1	2	3	4	5
Операційна система	Windows, Linux, Android, macOS	Windows, Linux, Android, macOS	Windows, Linux, Android, macOS	Windows, Linux, Android, macOS
Гнучкість / кастомізація	Обмежена	Обмежена	Висока	Дуже висока
Аутентифікація	Анонімний ID	Access key	Сертифікати / паролі	UUID
Захист від DPI	Середній	Середній	Середній	Високий
Підтримка VLess	-	-	-	+
Підтримка uTLS	-	-	-	+
Підтримка XTLS Vision	-	-	-	+

Для реалізації програмного комплексу буде використаний заздалегідь готовий клієнт з графічним інтерфейсом, що буде описаний у третьому розділі.

1.6 Висновки до розділу 1

У результаті аналізу теоретичних аспектів, поданих у цьому розділі, сформовано цілісне уявлення про те, як працює інтернет, як організовано обмін даними між користувачем і віддаленим ресурсом, та яку роль у цьому процесі відіграє інтернет провайдер користувача, ISP. Провайдер може виступати як активним учасником, так і посередником у впровадженні цензури, через блокування IP-адрес, аналіз заголовків пакетів та фільтрацію трафіку за вмістом з допомогою DPI. Наведений огляд дії цензури показує, що вони здатні виявляти навіть зашифрований або тунельований трафік за непрямими ознаками, такими як, частота, розмір пакетів, або типова поведінка протоколів. Це створює потребу в складніших підходах до маскування, зокрема шляхом обфускації трафіку. Обрано ручний метод аналізу трафіку, який буде проводитись за допомогою Wireshark. Розглянуто способи фільтрації трафіку, виділення окремих потоків, ідентифікації процесів-джерел, а також інструменти статистичної обробки, включно з ієрархією протоколів, аналізом розмірів пакетів та побудовою I/O графіків. Розглянуто два основні підходи обфускації трафіку, стеганографічний, що намагається імітувати дозволений трафік, та поліморфний, що прагне повністю приховати структуру трафіку. Стеганографічні методи менш надійні через труднощі точної імітації легітимних протоколів. Поліморфний підхід, натомість, є перспективнішим, особливо у поєднанні з повним шифруванням та відсутністю відкритих сигнатур у заголовках.

Проведений аналіз дозволив поставити мету роботи: створення програмного комплексу, що буде забезпечувати обфускацію трафіку. Для досягнення мети поставлені такі задачі: розробити набір технологій, що забезпечить надання VPN послуг користувачеві, та не буде відображати відомих ознак VPN технологій під час своєї роботи, побудувати програмний комплекс, що складається з серверної і клієнтської частини, та забезпечить обфускацію VPN трафіку.

2 УДОСКОНАЛЕННЯ ЗАСОБУ ОБФУСКАЦІЇ ТРАФІКУ

У цьому розділі досліджується технічна реалізація програмного комплексу для обфускації інтернет-трафіку з метою обходу цензури та забезпечення приватності користувача. Описано вибір технологій, інструментів і середовища розробки, а також детально проаналізовано специфіку сучасних протоколів V2Ray, зокрема VLess у поєднанні з XTLS-Vision. Було описано принципи побудови архітектури клієнт-серверної взаємодії, обробку конфігурацій у форматі JSON, а також моделювання структури системи засобами UML та блок-схем.

2.1 Обґрунтування удосконалення існуючих способів обфускації трафіку

Метою роботи є створення програмного комплексу для приховання VPN активності, зокрема на українських територіях які тимчасово під російською окупацією. Надалі, російську сторону контролю та спостереження за трафіком, що надходить від користувачів її ISP, буде приводитись під назвою цензор. Можна визначити який трафік цензор блокує, наприклад той самий VPN трафік або спроби обфускації, проте не можна визначити, який з цих трафіків він здатен розпізнати, оскільки про це не повідомляється відкрито з їхньої сторони, а також не повідомляється власне користувач, що такий трафік утворив. Саме цим і користується цензор, тому, базуючись на відкритих даних та дослідженнях протоколів обфускації, що доступні у використання будь-кому, було обрано зв’язку протоколів, що найкраще справляється з обфускацією VPN трафіку, далі буде проведено порівняння та обґрунтування цього вибору.

У минулому розділі було розглянуто існуючі технології обфускації, вони мають суттєві недоліки, що сприяє виявленню їх роботи у мережі, та створює небезпеку для користувача. Наприклад, підтримка протоколу Shadowsocks, його винахідником припинилась у 2015 році, після чого інші волонтери, продовжили його розвиток, випускаючи багато модифікованих версій протоколу, щоб підтримувати його ефективність. Було змінено оригінальну процедуру аутентифікації користувача, та останньою версією є Shadowsocks2022, після

досліджень було виявлено, що китайський цензор, зміг виявити трафік Shadowsocks за відношенням 1 до 0 у корисному навантаженні такого трафіку, який описано раніше, імовірно російський цензор, теж може проводити таке виявлення [10].

Далі, SSH тунель, що утворює зашифрований транспорт зазвичай для пакетів консольного керування, де найчастіше, пакет не перебільшує розміру в 100 байтів. Відповідно, передаючи дата чи відео трафік, характеристики якого були наведені у підрозділі 1.3, через проксування інкапсульоване у SSH тунель, отримаємо трафік, який статистично не схожий на трафік керування консоллю. Трафік керування консоллю, є рідким та має малий об’єм пакетів, відповідно, під час передачі відео, трафік є постійним та навантаженим, а під час передачі дати, трафік є також рідкісним, але перевантаженим по міркам консольного використання SSH, це є основна ознака виявлення обфускації за допомогою SSH [11].

VMess. Всі пакети, що проходять через VMess є одного розміру та структури, що і створює його перший недолік. Створені пакети передаються TLS для проведення транспорту, при цьому TLS буде мати ознаки, що він утворений нестандартним чином, про це буде йтися далі. У цій конфігурації VMess отримує вразливість до переліку атак, в ході вирішення яких, значення alterId стає 0, тобто зникає. А також, сервер при отримані запиту з невірним UUID значенням, немає чіткої відповіді, чим можуть скористуватись атаки активного зондування, VLess ці недоліки виправляє [12] [36].

Тому, для вирішення цієї проблеми з існуючими способами обфускації, було прийняте рішення про обрання найкращих за характеристиками технологій та об’єднати їх в єдиний програмний комплекс. Протокол VLess стане основою комплексу, через його характеристики заявлені автором та оцінки іншими користувачами. Це протокол проксування з обфускацією, що по суті, є наступною версією VMess, з виправленими недоліками попереднього, та активною підтримкою. Після надсилання на певну системну ланку SOCKS запиту, що утворила програма користувача, VLess додає до нього значення UUID, яке повинно бути узгоджене з VLess сервером перед під’єднанням. Перед передачею цих

пакетів до TLS, який забезпечить їх транспортування, вони всі випадково змінюються у розмірі, так званий паддінг, від англійського слова padding, що вирішує проблему виявлення яка присутня у VMess. Шифрування не входить у роботу протоколу, цим займається TLS, або інший транспорт, тому VLess працює швидше ніж його попередник, також, VLess імовірно почне використовувати стиснення пакетів перед їх відправкою алгоритмом Zstd [13]. Саме VLess буде додано у стек протоколів обфускації трафіку, для виконання роботи.

VLess працює тільки з транспортним протоколом прикладного рівня, що також, повинен забезпечувати шифрування. Транспортним протоколом прикладного рівня може стати Websocket, TLS або інші, проте вибір у розробці лишиться за TLS, оскільки він найбільш поширений з доступних транспортів, та його копії, що створюють для ефективної обфускації, активно розробляються волонтерами, так званою антицензурною спільнотою. Тому далі, будуть розглянуті проблеми використання TLS в обфускації та її вирішення. Також, необхідно додати, що для підключення до проксі сервера, користувач може використовувати CDN мережі, які проводять "активну" маршрутизацію трафіку, змінюючи IP-адресу призначення та джерела пакетів відповідно, що є досить корисним аспектом у реалізації обфускації, проте, CDN не можна користуватись у нижче наведених реалізаціях TLS, через його суворі дозволи, що до протоколів, які можуть по ньому подорожувати, та можна використовувати з іншими прикладними транспортними протоколами, обфускація через CDN у роботі не розглядатиметься [38].

TLS, у варіанті його виконання фреймворком V2Ray, має декілька проблем. Під час ініціації зв’язку між користувачем та сервером, першій пакети TLS, які називають пакетами рукостискання, містять певні відомості, необхідні для встановлення зв’язку, ці відомості, хоча і не мають прямого посилання на програму його реалізації, проте по специфічних наборах цих відомостей, можна встановити, чи TLS ініціацію утворено, наприклад браузером Chrome та його бібліотекою BoringSSL, чи GoLang з бібліотекою crypto/tls, на якій власне і створений фреймворк V2Ray, та відповідно ідентифікувати аномальну TLS ініціацію зв’язку [37]. Цими відомостями є набір шифрів, набір розширень, значення ALPN, та інші,

цей набір відомостей називається відбитками або англійський термін fingerprint, у різних виконань TLS він різний. Одним з рішень цієї проблеми є використання модернізованої бібліотеки для виконання TLS з'єднання під назвою uTLS, який на сьогоднішній день, успішно працює для протидії виявленню нестандартних TLS з'єднань. uTLS дозволяє, все ще використовуючи GoLang та crypto/tls, змінювати відбитки рукостискання або на випадкові, тим самим приховуючи походження TLS, або замінювати їх на відбитки найпопулярніших браузерів, таких як Chrome, Firefox та інші [14].

Проте, в uTLS не передбачено вирішення ще одної проблеми, а саме виявлення встановлення нового TLS з'єднання всередині вже існуючого, так званий TLS в TLS. Між проксі сервером та користувачем, передача відбувається TLS сесією, під час якої, користувач може встановити сесію та передавати пакети нового TLS зв'язку з кінцевим ресурсом, до якого він звертається через проксі сервер. Тим самим, у вже існуючому TLS транспорті з проксі, який має свої заголовки, додаються нові заголовки TLS пакетів сесії з кінцевим ресурсом, утворюючи наступну ознаку виявлення [15]: Чим більше шарів шифрування, тим важчим буде заголовок. Хоча це збільшення невелике, воно може бути більш очевидним для невеликих даних (таких як пакети відповідей), а деякі пакети можуть перевищувати ліміт MTU базової мережі. Найголовніше, що оскільки кожен пакет додає однакову довжину, він може мати певні статистичні характеристики. Та оскільки, TLS є досить популярним транспортом для прикладних протоколів, ця проблема може виникати часто. Тому необхідно застосувати технологію XTLS, зокрема, її одну з останніх версій XTLS-Vision, суть дії якої, це призупинення TLS сесії, коли XTLS виявляє, що користувач встановлює нову TLS сесію поверх TLS сесії з його проксі. Працює він наступним чином, коли програма користувача починає проводити TLS рукостискання, воно як і в схемі з простим TLS, проходить обробку VLess та надходить до TLS сесії з проксі, яка транспортує його до проксі сервера, де той, ретранслює його на кінцевий ресурс. Проксі отримує відповідь від кінцевого ресурсу, та відправляє її до користувача, і так до кінця рукостискання користувача та кінцевого ресурсу, тобто звичайна

взаємодія як і в TLS. Після цього [15]: проксі-клієнт користувача вставляє UUID у перший внутрішній зашифрований пакет TLS сесії користувача та кінцевого ресурсу, щоб повідомити проксі-серверу "Ми готові до роботи без ліцензії! ", після другого внутрішнього зашифрованого пакета XTLS не виконує жодних операцій з пакетом даних, а просто копіює його. Тобто, прибираючи наявну TLS сесію з проксі сервером, та пускаючи до проксі тільки встановлений, зашифрований TLS трафік сесії користувача та кінцевого ресурсу.

Відповідно, протоколами, за якими буде створюватися програмний комплекс обфускації трафіку стануть VLess + XTLS-Vision. Також, необхідно додати, що ця зв’язка може бути недосконалим рішенням з обфускації трафіку, проте, як вказали автори одної з останніх модифікацій Shadowsocks [16]: "Будьте більш толерантними до недосконалих рішень для обходу та не відмовляйтеся від недосконалого рішення занадто рано; …той факт, що будь-який цензор менш гнучкий, ніж спільнота антицензури, часто ігнорується. І через це багато рішень для обходу були відхилені та передчасно припинені просто тому, що вони "мають слабкі місця". Це часто відбувається тому, що коли розробники та дослідники антицензурних рішень уявляють себе цензорами, вони схильні зосереджуватися на першому кроці цензури – "попередньому дослідженні" – і думають, що рішення для обходу буде легко заблоковано, однак насправді, вони недооцінили тривалий процес фінансування заявок, офіційних наукових досліджень, розробки продукту, налагодження, опитування реального трафіку, експериментального розгортання, та нарешті, загальнонаціонального розгортання, з яким стикаються справжні цензори. Фактично, якщо розробник антицензурних рішень витрачає день на розгортання нового інструменту проти цензури, цензору потрібно багато часу, енергії, людських, матеріальних та фінансових ресурсів, щоб заблокувати його, що може зайняти багато місяців, тому, цей "недосконалий" інструмент може слугувати чудовим важелем впливу. ".

2.2 Опис середовища розробки, мови програмування

Створення програмного комплексу з обфускації трафіку, відбувалось за допомогою різних мов програмування та середовищ розробки, проведемо їх огляд з поясненням їх ролей.

Спершу розглянемо Golang. Go – це високорівнева мова програмування загального призначення , яка статично типізується та компілюється . Вона відома простотою свого синтаксису та ефективністю розробки, яку забезпечує включення великої стандартної бібліотеки, що задовольняє багато потреб для поширених проектів. Вона була розроблена в Google у 2007 році Робертом Грізмером , Робом Пайком та Кеном Томпсоном, та публічно оголошена в листопаді 2009 року. Синтаксично вона схожа на C , але також має безпеку пам'яті , збирання сміття , структурну типізацію та паралелізм у стилі CSP . Її часто називають Golang , щоб уникнути двозначності та через її попереднє доменне ім'я [19]. У розробці проекту, використання мови програмування Golang відіграє ключову роль, оскільки саме на ній написано ядро системи проксування та тунелювання – XRay-core. Go обрано як основну мову для бекенд-частини клієнтського додатку, через її високу продуктивність, простоту управління, а також, через зручну підтримку мережевих операцій, що є критично важливим для реалізації мережевих протоколів, таких як VLess, VMess, XTLS тощо. Golang також компілюється в статично лінковані бінарники, що спрощує розгортання на різних системах, включаючи сервери без повноцінного середовища користувача.

Далі, розберемо фреймфорк Qt мови програмування C++. Qt – крос-платформовий інструментарій розробки програмного забезпечення мовою програмування C++. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем, просто компілюючи текст програми для кожної операційної системи без зміни початкового коду [39]. Містить всі основні класи, які можуть бути потрібні для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML. Бібліотека дозволяє керувати потоками, працювати з мережею та забезпечує крос-платформовий доступ

до файлів [20]. Та саме графічна оболонка клієнта користувача, реалізована за допомогою C++ у поєднанні з фреймворком Qt. Його було обрано завдяки його стабільності, широкій підтримці платформ, багатому інструментарію для побудови GUI, та підтримці мовою C++, яка надає високий контроль над ресурсами та продуктивністю.

Конфігураційні файли для V2Ray, як на сервері, так і клієнті, описані у форматі JSON. JavaScript Object Notation, JSON – це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, що може бути прочитаний людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передавання структурованої інформації через мережу, завдяки процесу, що називають серіалізацією [21]. Його було обрано через простоту, зрозумілу структуру ключ-значення, відсутність надмірної формальності, як у XML, та широке розповсюдження, JSON підтримується практично всіма мовами програмування.

Для розгортання серверної частини проекту було обрано операційну систему Linux, зокрема Ubuntu, та було орендовано послуги VPS Digital Ocean, з виділеної IP-адреси, обчислювальних ресурсів та цифрової пам’яті. Ubuntu це дистрибутив Linux для робочих станцій, лептопів і серверів, найпопулярніший у світі дистрибутив Linux. Серед основних цілей Ubuntu – надання сучасного й водночас стабільного програмного забезпечення для пересічного користувача із сильним акцентом на простоту встановлення та користування. Серверний варіант системи включає також засоби, потрібні для організації сервера баз даних, вебсервера, сервера електронної пошти тощо [22]. Linux Ubuntu було обрано через її стабільність, широке розповсюдження у сфері серверного хостингу, наявність багатьох інструментів для адміністрування, пакетного менеджера apt, зручності для автоматизації та налаштування сервісів. Ubuntu має велике спільноту користувачів, змістовну документацію, і сумісність з більшістю VPS-провайдерів, що робить її практичним вибором для мережевих застосунків і сервісів з підтримкою systemd [41].

2.3 Проектування UML діаграм послідовностей та блок-схеми роботи програмного комплексу

У підрозділі буде складено дві UML діаграми послідовностей, для двох сценаріїв роботи програмного комплексу, з блок-схемою алгоритмів роботи.

UML-діаграма послідовностей, Sequence Diagram – це тип діаграми в Unified Modeling Language, що використовується для моделювання взаємодії об’єктів у часі. Вона показує, як об’єкти обмінюються повідомленнями між собою для виконання певного сценарію або операції. UML-діаграма складається з: об'єктів, що є елементами, які беруть участь у взаємодії, з життєвих ліній, що позначаються вертикальними пунктирними лініями, які показують "життя" об'єкта протягом сценарію, з повідомлень, що є горизонтальними стрілками між життєвими лініями, що відображають виклики методів або обмін інформацією, повідомлення відповідей позначаються пунктирними лініями. Ці діаграми стануть корисними для відображення логіки взаємодії між компонентами системи програмного комплексу. Приведемо першу UML діаграму роботи системи, коли програма користувача не ініціює TLS зв’язок, рисунок 2.1. Ця діаграма відображає, як відбувається зв’язок та передача даних при роботі користувача з проксі сервером в межах роботи програмного комплексу з обфускації трафіку, який ініціює програма, App, користувача [40].

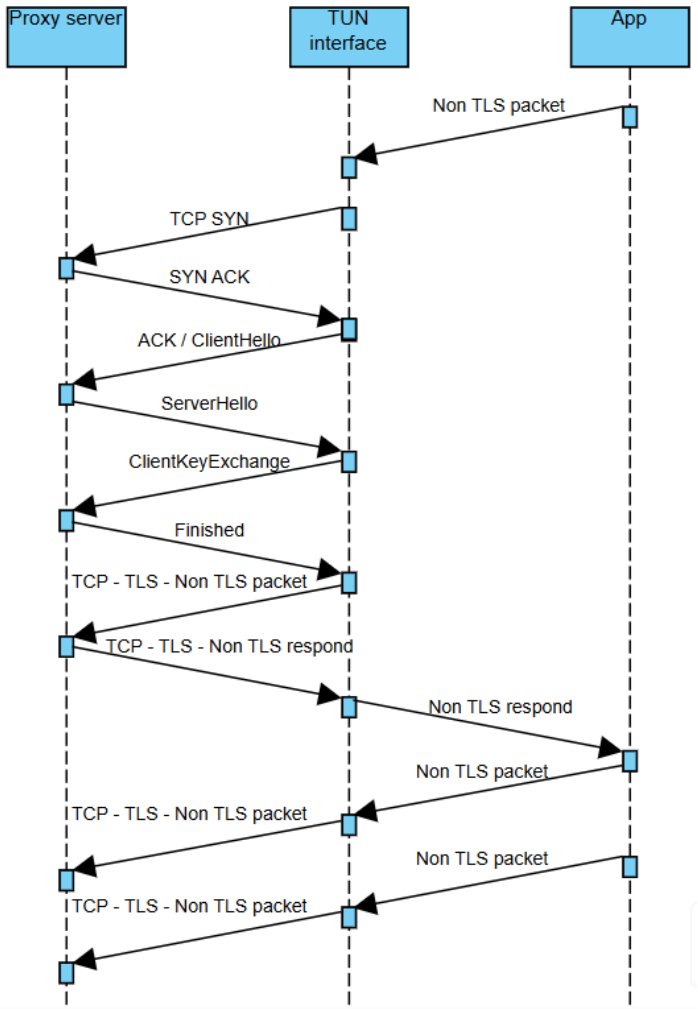


Рисунок 2.1 – UML діаграма роботи системи при стандартній взаємодії

Крок 1. Програма користувача, утворює запит на прикладному протоколі, що не є TLS запитом, Non TLS, та надсилає його до віртуального мережевого інтерфейсу, TUN, згідно правил маршрутизації, що встановлює VLess.

Крок 2. При надходження на TUN, пакет з програми, починає очікувати, поки встановиться TCP та поверх неї, TLS сесія з проксі сервером. Це відбувається шістьма пакетами, від TCP SYN до Finished.

Крок 3. Після чого, по справжньому TLS з’єднанню з проксі, користувач починає передавати пакети, змінені VLess, інкапсулювані у корисне навантаження цього TLS. Власне, після пакету finished, перший пакет програми користувача, буде переданий у TLS тунель. Після встановлення TLS тунелю з користувачем, проксі, у першому ж пакеті, що несене TLS корисне навантаження, перевіряє UUID клієнта, якщо той не підходить – термінує або перенаправляє зв’язок. Та перед тим, як пакет

програми надсилається з TUN до проксі, VLess додає до нього UUID та проводить зміну його розміру, паддінг. Відповідно UUID додається тільки до першого пакету TCP – TLS – Non TLS Packet, та більше, ні до яких наступних.

Крок 4. В результаті, починає працювати зв’язок між програмою користувача та проксі, обмінюючись повідомленнями запитів та відповідей. Також, встановлення сесії користувача з проксі відбувається з урахування зміни його відбитків. Розглянемо наступну діаграму, що включає встановлення програмою користувача TLS з’єднання з кінцевим ресурсом, рисунок 2.2.

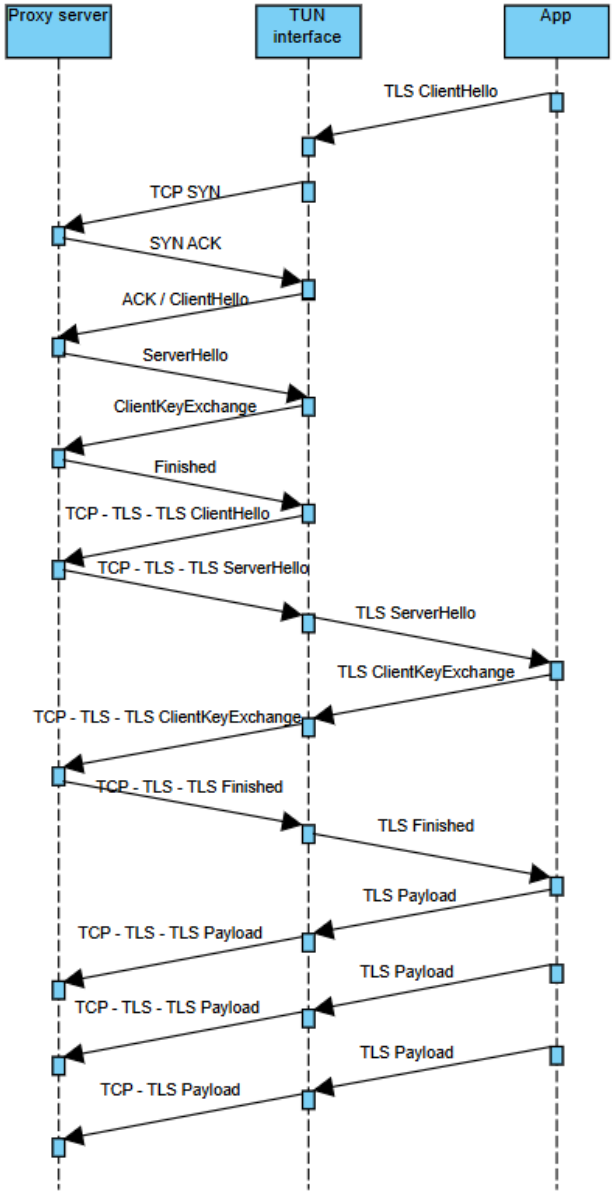


Рисунок 2.2 – UML діаграма роботи системи при виконанні протоколу TLS

Проксі сервер завжди спостерігає за трафіком, який надходить на нього від користувача, та під час встановлення TLS сесії між програмою користувача та кінцевим ресурсом, XTLS-Vision забезпечує припинення TLS шифрування між користувачем та проксі, тим самим прибираючи проблему виявлення TLS в TLS, що була описана в підрозділі 2.1. На безпеку це не впливає, оскільки пакети, що покидають користувача, все одно є зашифрованими TLS сесією програми користувача та кінцевого ресурсу.

Крок 1. Процес підключення відбувається схожим чином, що і в попередньому варіанті, з першим TCP – TLS – TLS ClientHello пакетом, передається значення UUID для аутентифікації користувача проксі сервером, а також пакет TLS ClientHello рукостискання, що проводить програма користувача та кінцевий ресурс, який передається, інкапсульованим, у TCP TLS з'єднання між користувачем та проксі. Всі наступні пакети TLS рукостискання не будуть мати UUID значення.

Крок 2. Так відбувається, до поки, на TUN не приходить перший TLS Payload пакет сесії програми користувача та кінцевого ресурсу, куди VLess знову, поміщає UUID значення.

Крок 3. Проксі сервер, коли отримає цей TLS Payload пакет, перевіряє чи він містить значення UUID.

Крок 4. Якщо так – через один пакет після цього, TLS тунель між користувачем та проксі буде прибрано, та буде залишено лише TCP сесію, по якій будуть проходити TLS Payload пакети сесії програми користувача та кінцевого ресурсу – останній пакет TCP – TLS Payload. Значення UUID для сигналізації про початок XTLS сесії додається тільки до першого пакету TLS Payload.

Також, у такому сюжеті не може працювати технологія MUX, що дозволяє, над одним TCP з'єднання передавати декілька сесій спілкування прикладних протоколів, тому що, якщо прикладний протокол не передбачає шифрування або анонімізацію, це створює порушення конфіденційності під час передачі.

Далі, буде приведена блок-схема роботи програмного комплексу з обфускації трафіку, що являє собою графічне зображення алгоритму або процесу, що представляє послідовність дій за допомогою фігур, блоків, з'єднаних стрілками.

Кожен з цих блоків виконує певну функцію, еліпси вказують на початок та кінець програми, прямокутники позначають операції або дії що будуть проводитись, ромби перевіряють умови, паралелограми виводять дані у формі, придатній до обробки. Стрілки вказують напрямок потоку виконання. Блок-схеми допомагають зрозуміти логіку роботи програмного комплексу, спростити аналіз, налагодження та документування системи, рисунок 2.3.

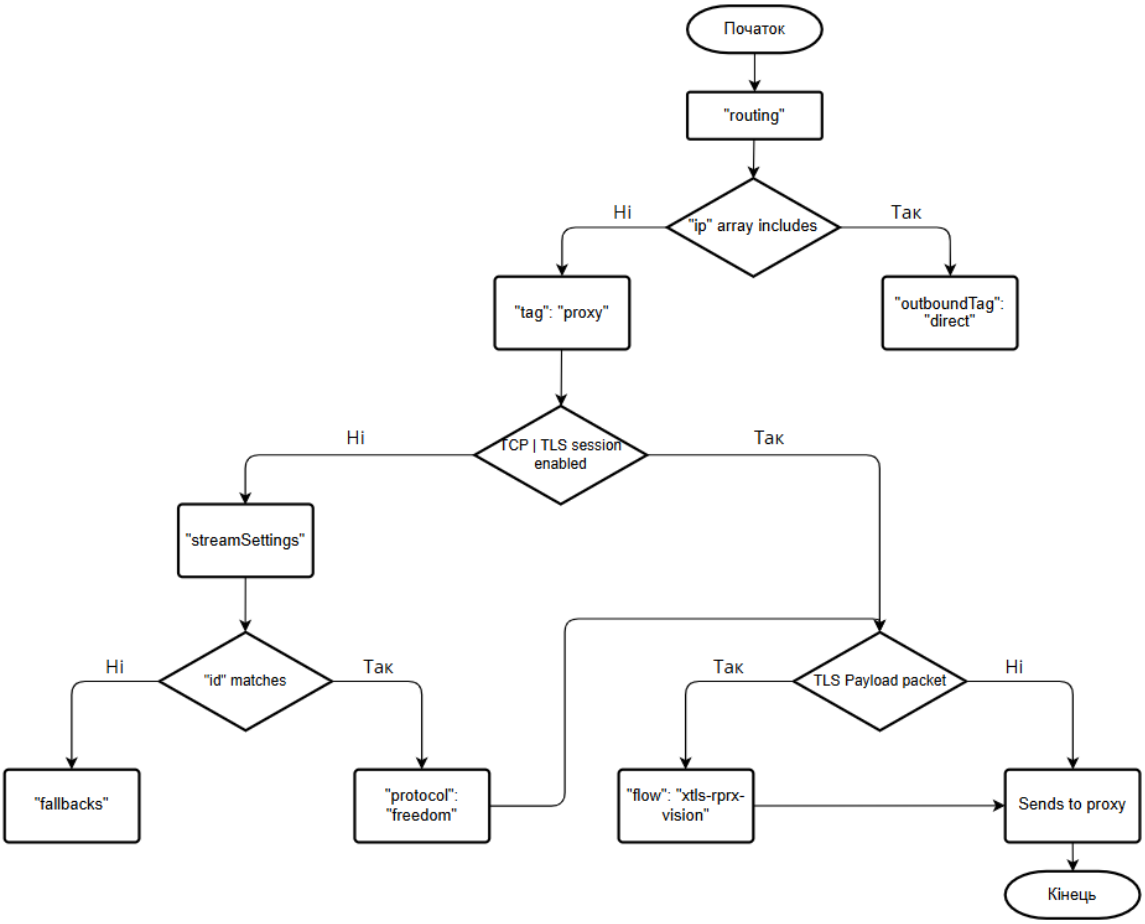


Рисунок 2.3 – Блок-схема роботи програмного комплексу

Крок 1. Початок.

Блок-схема описує логіку обробки пакету, коли той створюється програмою користувача, схема поєднує сторону клієнта та сервера.

Крок 2. Коли програма створює пакет прикладного протоколу, він, по дії "routing" у файлі конфігурації V2Ray на клієнті, перевіряється на умову "ip" array includes.

Крок 3. При якій, якщо IP-адреса ресурсу, до якого звертається програма користувача є у цьому масиві, цей пакет, прямо перенаправляється у "відкритий" інтернет, без проксування дією "outboundTag":"direct". Це створено тому, що існує загроза виявлення наявності обфускованого трафіку, що надходить від користувача, цензором. Загроза полягає в наступному, якщо під час використання проксі, користувач переходить до російських сайтів, зазвичай через TLS, вони можуть мати вбудовані алгоритми відслідковування походження користувача, які працюють у зв’язці з ISP користувача, та відповідно наприклад, коли в певний момент, користувач починає ініційовувати сесію з таким сайтом та обмінюватись з ним пакетами, час та кількість пакетів, спершу, фіксуються його ISP та передаються далі, вищому рівнем цензору, який порівнює ці дані з даними підключення та спілкування відвідувачів їхнього сайту, та оскільки мережева взаємодія працює в межах мілісекунд, такий користувач, може бути ідентифікований як відвідувач такого сайту, проте чомусь, цей користувач не звертався на IP-адресу цього сайту безпосередньо, а власне, до свого проксі, тут і виникає проблема. Тому, необхідно буде внести такі сайти до "ip" array, щоб користувач звертався до них напряму, не через проксі.

Крок 4. Якщо ж IP-адреса до якої звертається програма користувача не входить в "ip" array, то пакет перенаправляється на "tag":"proxy", власне на TUN інтерфейс, де відбувається перевірка, чи для цього пакету вже створена працююча TCP TLS сесія з проксі сервером.

Крок 5. Якщо ні – утворюється TCP TLS сесія з проксі, описана у UML діаграмах раніше, дією "streamSettings". Після чого у першому пакеті TLS Payload цієї сесії, сервер перевіряє чи сходиться UUID користувача, з тим який встановлений в його конфігурації у значенні "id".

Крок 6. Якщо ні – сервер перенаправляє підключення на інший сайт, або відправляє свою, підставну, HTML сторінку.

Крок 7. Якщо так, сесія TCP TLS між користувачем та проксі вважається успішною, тому повертаємось на етап TCP TLS session enabled.

Крок 8. Та переходимо до наступної перевірки – чи пакет, який надійшов від програми до TUN, є TLS Payload.

Крок 9. Якщо так, то починається процес XTLS-Vision, який направлений на прибирання TLS сесії між проксі та користувачем, після чого пакет надсилається до проксі, Sends to proxy,

Крок 10. Кінець. Та якщо пакет не є TLS Payload, він одразу надсилається до проксі.

2.4 Висновки до розділу 2

У межах другого розділу було розроблено технічне підґрунтя для створення програмного комплексу, що забезпечує обфускацію інтернет-трафіку та мінімізацію ймовірності його блокування системами DPI. Здійснено обґрунтування вибору мови програмування Go для реалізації серверної частини, фреймворку Qt/C++ через його кросплатформеність, гнучкість і зручну інтеграцію з низькорівневими мережевими викликами, для створення графічного інтерфейсу користувача, а формат конфігурацій було створено на основі JSON, що спрощує збереження, обробку і передачу налаштувань як на стороні клієнта, так і сервера. У процесі проектування було враховано підтримку UUID-ідентифікації, мультиплексування з'єднань, MUX, шифрування трафіку, підтримку різних протоколів транспортного рівня, TLS, XTLS і гнучку маршрутизацію запитів. Для візуалізації архітектури й аналізу логіки взаємодії між компонентами розроблено кілька UML-діаграм, а також, побудовано блок-схему, яка демонструє основні етапи ініціалізації з'єднання, перевірки конфігурації, маршрутизації пакетів та передачі даних через тунель. Усі ці моделі сприяють кращому розумінню внутрішньої логіки системи та забезпечують основу для подальшої розробки, тестування й підтримки проекту. Таким чином, у цьому розділі закладено основу для практичної реалізації системи обфускації, яка надає стійкість до аналізу трафіку та гнучку архітектуру, адаптовану до мереж із активною цензурою.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

У цьому розділі розглянуто процес практичної реалізації програмного комплексу з обфускації трафіку, розробленого на основі попереднього теоретичного аналізу. Детально описано поетапне налаштування як серверної, так і клієнтської частин, а також проведено аналіз отриманого результату шляхом порівняння трафіку при звичайному, VPN і обфускованому з'єднанні. Розділ має на меті показати не лише працездатність запропонованого рішення, а й ефективність його застосування для приховування ознак VPN-активності.

3.1 Налаштування серверної частини програмного комплексу

Створення програмного комплексу з обфускації трафіку буде включати налаштування віддаленого проксі сервера та клієнтського додатку на стороні користувача. На стороні проксі сервера буде встановлений та налаштований консольний клієнт, core, за допомогою командної оболонки Bash системи Linux Ubuntu. Відповідно, проксі сервер налаштовується тільки для логічної обробки вхідних пакетів, без графічного інтерфейсу взаємодії, GUI. Було орендовано послугу поширених ресурсів CPU та інших ресурсів разом з IP-адресою, на якій буде працювати проксі сервер програмного комплексу. Провайдером віртуального сервера, VPS, було обрано Digital Ocean, через його доступність, цінову політику та зручну у користуванні панель керування віртуальними машинами. Також, у роботі програми виконується встановлення TLS сесії між користувачем та проксі сервером, для утворення тунелю передачі, тому, необхідно мати у власності певне доменне ім'я, якому необхідно отримати сертифікат TLS, від центру сертифікації, що спеціалізується в цьому. Відповідно, було придбане доменне ім'я під назвою brpalukh.space, з топ доменом .space, та під час отримання сертифікату використовувався центр сертифікації Let's Encrypt. Віддалений сервер створений на системі Linux Ubuntu 24.10, та отримавши доступ до нього, переходим до його налаштування. Перші два підрозділи третього розділу можна використати у якості

інструкції для розгортання власного, аналогічного, програмного комплексу з обфускації трафіку.

Спочатку були проведені наступні команди, перша оновлює перелік доступних до встановлення пакетів послуг для системи, а друга оновлює до останніх версій вже встановлені пакети послуг:

```
~# apt update
~# apt upgrade
```

Після цього, було проведено команду завантаження V2Ray фреймворку з репозиторію Github, який необхідно буде розпакувати з архіву:

```
~# wget https://github.com/XTLS/Xray-core/releases/download/v25.4.30/Xray-linux-64.zip
```

Наступними командами, було створено окрему директорію для розміщення в ній фреймворку та розпаковано в неї .zip файл, де прапорець -p забезпечує створення всіх необхідних батьківських каталогів в процесі створення, якщо вони не створені, а -d вказує шлях, куди буде розпакований файл:

```
~# mkdir -p /server/xray
~# unzip ./Xray-linux-64.zip -d /server/xray
```

В системі вже встановлена утиліта certbot, що при виклиці, забезпечує автоматичне отримання сертифікату TLS, для вашого домену, через центр сертифікації Let’s Encrypt, тому одразу було проведено її:

```
~# certbot certonly --standalone --preferred-challenges http -d brpalukh.space
```

Вивід команди вказує, у яку директорію будуть поміщенні сертифікат сервера та його приватний ключ:

```
Certificate is saved at: /etc/letsencrypt/live/brpalukh.space/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/brpalukh.space/privkey.pem
```

Далі, було запущено службу V2Ray, та за допомогою встановленого текстового редактора Nano, створено та відкрито його файл конфігурації, у .json форматі:

```
~# nano /server/xray/config.json
```

Необхідно додати, що низькорівневі дії з трафіком, фреймворк V2Ray, проводить за своїми внутрішніми функціями, що вже налаштовані, та саме

керування цими функціями проводиться за допомогою конфігураційного файлу config.json. Фреймворк містить встановлений XTLS-Vision, тому він також налаштовується у config.json. Розберемо основні правила роботи проксі сервера, встановлені у конфігураційному файлі. Значення loglevel, визначає які рівні повідомлень, що системно утворює проксі сервер, будуть логуватись, зберігатись. "warning" означає, що будуть логуватися лише попередження, помилки і критичні події, але не всі повідомлення, щоб робив параметр "info":

```
"log": {
"loglevel": "warning"
},
```

Далі, йде налаштування обробників вхідних підключень, inbound. Сервер прослуховує на порту 443, протокол VLess, та якщо UUID у підключенні не сходиться, підключення, умовою fallbacks, перенаправляється на локальний хост, що прослуховує на порту 8002. Також, у приймачі, задається умова використання XTLS-Vision. Та при бажанні, можна додати декілька потенційних клієнтів, що можуть під'єднатись до проксі за допомогою свого UUID, в конфігурації наведений тільки один UUID:

```
"inbounds": [
{
"port": 443,
"protocol": "vless",
"settings": {
"clients": [
{
"id": "188fc747-0a10-42d1-8501-870194e015db",
"flow": "xtls-rprx-vision"
}
],
"decryption": "none",
"fallbacks": [
{
"dest": "8002",
"xver": 1
```

Далі, присутні налаштування підключення між проксі і користувачем. Підключення встановлюється за допомогою TCP та TLS сесії, при цьому значення SNI, Server Name Indication, повинно бути відомим для серверу, у цьому випадку brpalukh.space, мінімальна версія TLS протоколу що встановлюється, повинна бути 1.2, типом протоколу, імітація якого, встановлюється над TLS сесією є http/1.1 та

h2, зазначаються шляхи до даних сертифікату та значення ocspStapling вказує на те, як часто V2Ray буде кешувати OCSP-відповідь про статус TLS-сертифіката, що прискорить з’єднання користувачів до нього, та зменшить навантаження на OCSP сервери:

```
"streamSettings": {
"network": "tcp",
"security": "tls",
"tlsSettings": {
"rejectUnknownSni": true,
"minVersion": "1.2",
"alpn": [
"http/1.1",
"h2"
],
"certificates": [
{
"ocspStapling": 3600,
"certificateFile": "/etc/letsencrypt/live/brpalukh.space/fullchain.pem",
"keyFile": "/etc/letsencrypt/live/brpalukh.space/privkey.pem"
```

У конфігурації присутній тільки один обробник вихідних підключень, що здійснює підключення у разі проходження попередніх умові до "відкритого інтернету", freedom. Та керування політиками, де handshake: 2, визначає максимальний час у секундах, який дозволено витратити на встановлення початкового TLS з’єднання між користувачем і сервером, і connIdle: 120, визначає максимальний час у секундах, протягом якого з’єднання може не передавати жодних даних:

```
"outbounds": [
{
"protocol": "freedom",
"tag": "direct"
}
],
"policy": {
"levels": {
"0": {
"handshake": 2,
"connIdle": 120
```

Далі, було налаштовано проведення доменної адресації замість адресації за IP-адресою до кінцевого ресурсу, до якого звертається користувач. enabled": true вмикає цю функцію, що полягає у пошуку доменного імені при отримані пакетів від користувача та проведення доменної адресації, що сприяє кращому контролю та точності при створенні підключення до кінцевого ресурсу, оскільки одне доменне

ім’я може мати декілька IP-адрес. А destOverride, обирає під час виявлення яких протоколів необхідно виконувати цю функцію:

```
"sniffing": {
"enabled": true,
"destOverride": [
"http",
"tls"
```

На останок, було налаштовано локальний сервер nginx, що буде слухати адресу локального хосту, на порту 8002, у разі переведення вхідного з’єднання на умову fallback. Після встановлення nginx, створено та налаштовано його конфігураційний файл:

```
~# apt install nginx
~# nano /etc/nginx/sites-enabled/default
```

Сервер nginx, слухає як на IPv4 та і на IPv6 адресі локального хосту. При переведенні підключення до сервера, воно спочатку, згідно умови location, спробує надати підключенню index.html файл, якщо його не буде, URL посилання на певний сайт, та якщо його немає, відповість помилкою 404:

```
server {
listen 127.0.0.1:8002 default_server;
listen [::1]:8002 default_server;
root /var/www/html;
index index.html;
server_name _;
location / {
try_files $uri $uri/ =404;
```

Проведено перезапуск служби V2Ray та nginx, після чого, проксі сервер готовий до використання:

```
~# systemctl restart nginx
~# systemctl restart xray
```

V2Ray власноруч встановлює прослуховування на портах, які були задані у його файлі конфігурації.

3.2 Налаштування клієнтської частини програмного комплексу

Налаштування мережевої взаємодії на стороні користувача відбувалось за допомогою графічного клієнту Nekoray. Файл конфігурації .json на клієнті майже однаковий, як і на сервері, проте, необхідно продемонструвати структуру роботи

одного алгоритму при ній, що забезпечує захист від загрози, описаній у підрозділі 2.4, під малюнком 2.3. А саме, при спробі звернення програми користувача до IP-адреси, що є у списку ip, це підключення проводиться через "пряме", "outboundTag": "direct", підключення, без використання проксі:

```
"routing": {
"rules": [
{
"type": "field",
"ip": [
"104.26.8.59",
"104.26.9.59",
"172.67.75.163"
],
"outboundTag": "direct"
```

Також, вікно налаштування цієї функції наведено на рисунку 3.3.

Проксування від клієнту до сервера може відбуватися двома шляхами, перший, це коли програма користувача звертається на адресу локального хосту та другий, це коли програма звертається на віртуальний мережевий інтерфейс, на яких вже і відбувається зміна пакетів, для забезпечення проксування. Перший варіант називається системним проксі, та він має дві проблеми, коли програма користувача проводить DNS запит, вона може проводити його, без перенаправлення на локальний хост, а одразу на основний мережевий інтерфейс, а вже потім починає надсилати свої пакети до локального хосту, таким чином, ISP отримує можливість дізнатись, до яких сайтів бажає звернутись користувач. Та друга проблема, не всі програми користувача можуть підтримувати роботу, з переадресацією на локальний хост. Тому, для виконання роботи використовувалась, переадресація на віртуальний мережевий інтерфейс, TUN, де вже і діє VLess та XTLS. Інтерфейс клієнту зображений на рисунку 3.1, та містить, вибір TUN чи системного проксі, налаштування TUN та маршрутизації у вікні Preferences, можливість додавати нові профілі налаштування клієнта для проведення проксування у вікні Server, основне вікно зі вже створеними профілями, вікно з логуванням від системи користувача нижче та загальну інформацію про підключення. Почати та припинити сесію проксування можна правим натиском миші по профілю, який необхідно застосувати, та вибором опції Start та Stop, відповідно.

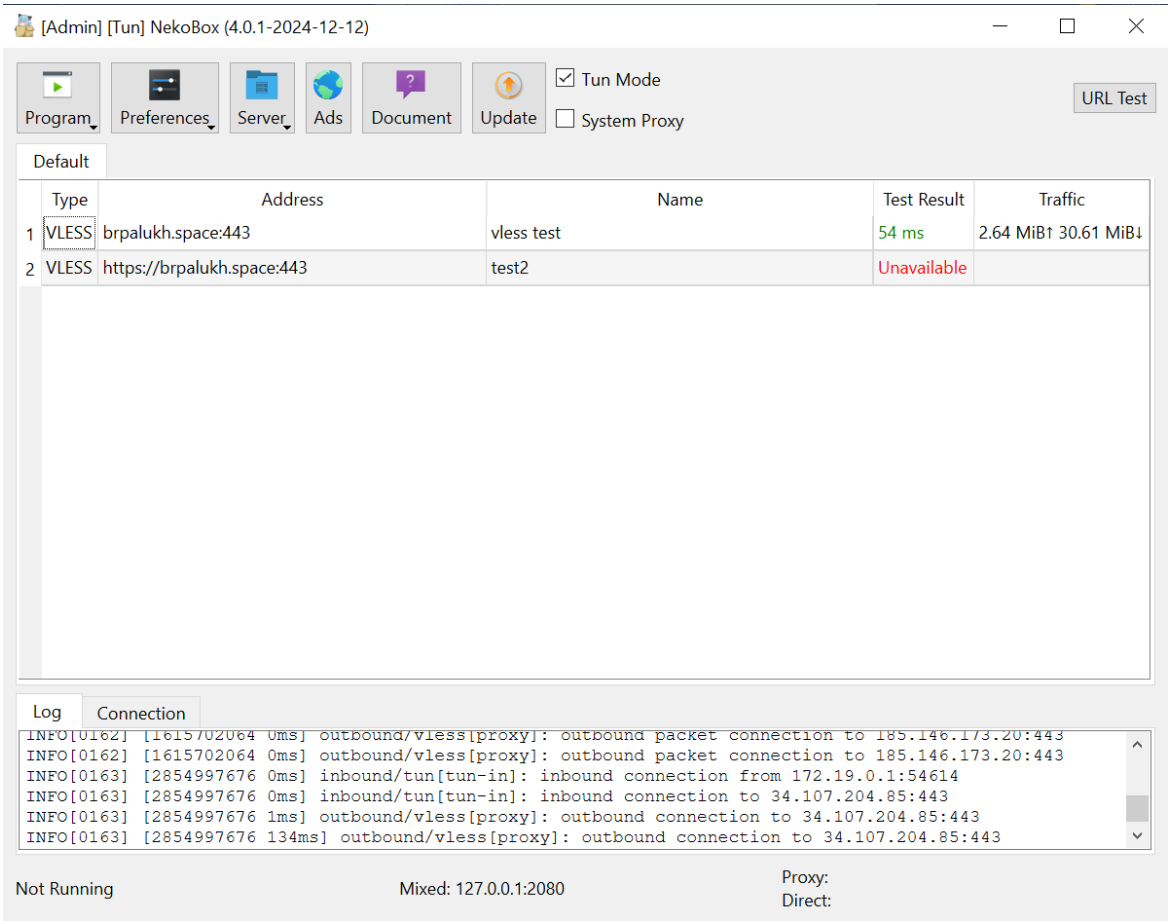


Рисунок 3.1 – Інтерфейс клієнту Nekoray

На рисунку 3.2, показані мінімальні налаштування для проведення проксі підключення, що були застосовані у реалізації. При них необхідно задати, домен проксі сервера, до якого планує звертатись користувач, порт, значення UUID, яке повинно відповідати цьому ж значенню у V2Ray конфігурації сервера, якщо передбачено – використання XTLS технології, тип підключення до серверу та для позбавлення відбитків нестандартного TLS – під який браузер слід замаскуватись, та при бажанні додати підставне значення SNI та ALPN.

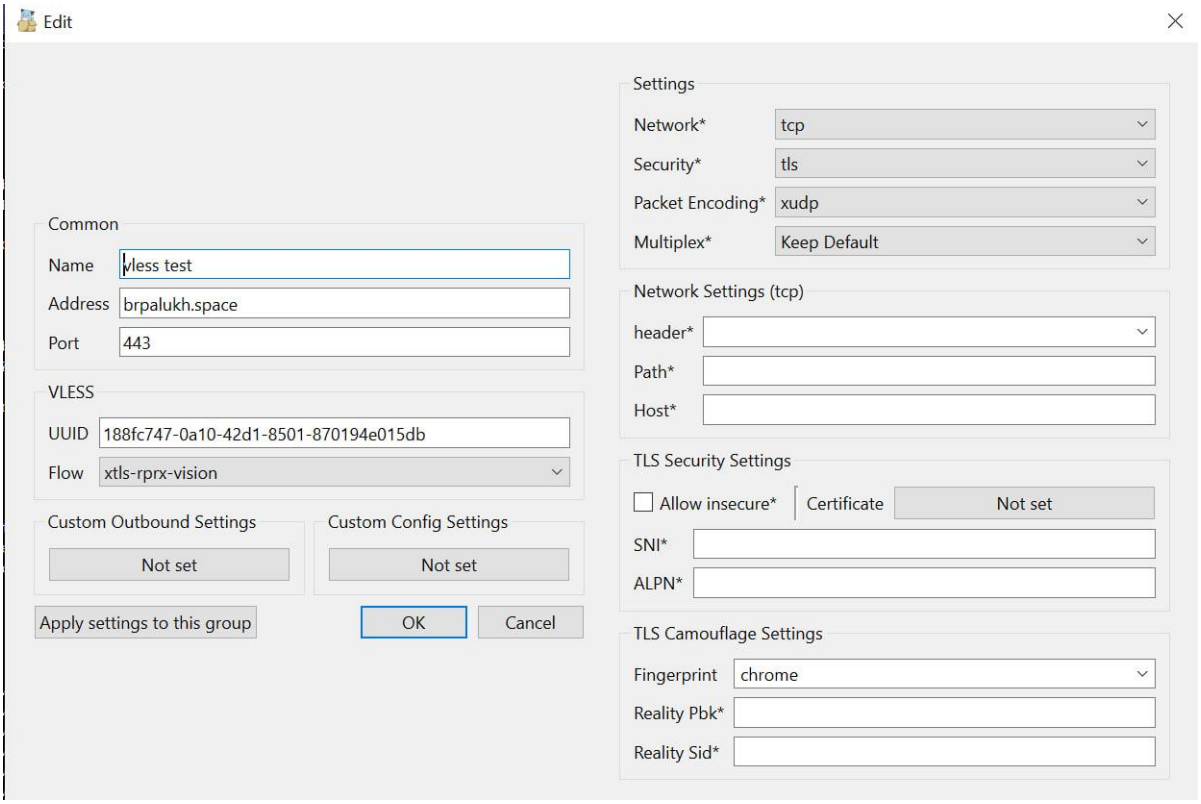


Рисунок 3.2 – Створення профілю проксування у Nekoray

Описаний раніше, алгоритм адресації, при якому, якщо програма користувача звертається на IP-адресу з відповідного списку, або доменне ім’я, це підключення переводиться у режим "direct", тобто не надсилається на проксі, а на звичайну маршрутизацію. На рисунку 3.3 зображено цю фільтрацію на доменне ім’я myip.com, сайт якого, відображає IP-адресу, з якої здійснюється підключення, для доведення працездатності роботи адресації.

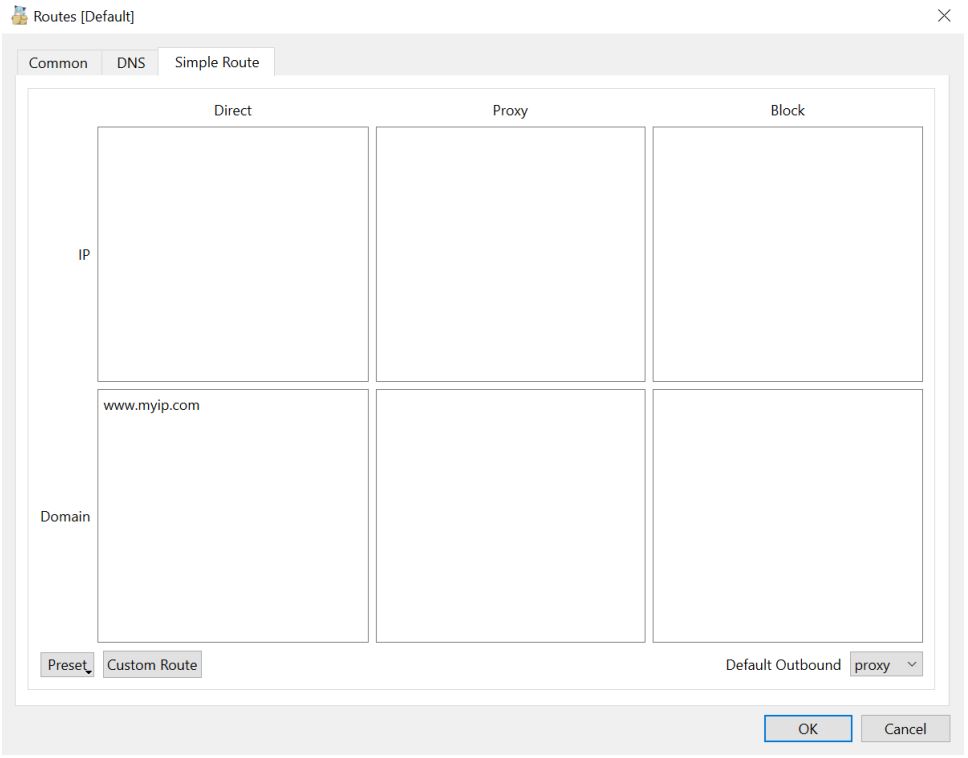


Рисунок 3.3 – Налаштування правил маршрутизації трафіку на клієнті

Після цих налаштувань, програмний комплекс з обфускації трафіку вважається готовим, у наступних підрозділах буде наведений аналіз його ефективності.

3.3 Аналіз та порівняння звичайного та VPN трафіку з обфускованим трафіком

Для аналізу будуть застосовані відомості з підрозділу 1.3 та програма Wireshark. Спершу буде наведений аналіз стандартного трафіку, без VPN технології, що потрібно буде для порівняння з обфускованим трафіком на схожість, та потім буде наведений аналіз трафіку з використанням VPN технології, із зазначенням як вона виявляється, який теж буде порівняно з обфускованим трафіком на зниклі ознаки. В якості трафіку, буде використовуватись підключення до вебсайтів, через протокол TLS, тобто data трафік. Будуть порівнюватися ознаки трафіку при ініціації з’єднання, та далі, у процесі передачі корисного навантаження, в кінці буде складена відповідна табличка. При аналізі, для зручної ідентифікації відповідного діалогу IP-адрес у вікні відображення потоку, буде застосовуватись

значення фільтру ip.stream eq __, де необхідно буде вказати номер діалогу, потоку, між IP-адресами, а щоб дізнатись цей номер, необхідно буде запустити вікно опції Statistics та Conversations, та за допомогою значень Packets, ідентифікувати діалог з найбільшою кількістю пакетів, найімовірніше, це і буде потрібний для дослідження діалог.

Перше підключення, зображене на рисунку 3.4, проводиться без використання VPN або обфускації, та відбувається наступним чином, отримавши DNS відповідь за IP-адреси, до яких програма користувача може звернутись, вона обирає першу та проводить встановлення TCP сесії з цією IP-адресою, рядок 26, в якому вже зазначається IP-адреса кінцевого ресурсу, є відкритою, тобто створює ризик порушення конфіденційності для користувача, через що, необхідно застосувати технологію проксування. Далі TCP сесія встановлюється у поєднанні з TLS, для пришвидшення та оптимізації підключень, при якому знову, відкрито, зазначається доменне ім’я до якого бажає звернутися користувач. Після встановлення сесії, всі дані передаються зашифровано, пакетами Application Data, маючи різний розмір, packet lengths. При цьому, до передачі корисного навантаження відбувається обмін, в середньому шістьма пакетами, після чого, трафік передається зі зворотнім надсиланням пакетів підтвердження, ACK, з різними інтервалами часу, рядки 100, 104, 113, 132 рисунку 3.5.

No.	Time	Source	Destination	Protocol	Length	Comment	Info
26	2.814071	192.168.0.104	104.26.8.59	TCP	66		50789 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
27	2.839245	104.26.8.59	192.168.0.104	TCP	66		443 → 50789 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM WS=8192
28	2.839317	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0
29	2.839917	192.168.0.104	104.26.8.59	TCP	1454		50789 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=1400 [TCP PDU reassembled in 30]
30	2.839917	192.168.0.104	104.26.8.59	TLSv1.3	408		Client Hello (SNI=www.myip.com)
31	2.856401	104.26.8.59	192.168.0.104	TCP	54		443 → 50789 [ACK] Seq=1 Ack=1755 Win=73728 Len=0
33	2.872592	104.26.8.59	192.168.0.104	TLSv1.3	1514		Server Hello, Change Cipher Spec
34	2.872787	104.26.8.59	192.168.0.104	TCP	1514		443 → 50789 [ACK] Seq=1461 Ack=1755 Win=73728 Len=1460 [TCP PDU reassembled in 35]
35	2.872787	104.26.8.59	192.168.0.104	TLSv1.3	592		Application Data
36	2.872826	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=1755 Ack=3459 Win=131584 Len=0
37	2.877369	192.168.0.104	104.26.8.59	TLSv1.3	118		Change Cipher Spec, Application Data
38	2.877657	192.168.0.104	104.26.8.59	TLSv1.3	146		Application Data
39	2.877954	192.168.0.104	104.26.8.59	TLSv1.3	540		Application Data
40	2.903016	104.26.8.59	192.168.0.104	TCP	54		443 → 50789 [ACK] Seq=3459 Ack=1911 Win=73728 Len=0
41	2.907778	104.26.8.59	192.168.0.104	TLSv1.3	575		Application Data, Application Data
42	2.908065	192.168.0.104	104.26.8.59	TLSv1.3	85		Application Data
43	2.927097	104.26.8.59	192.168.0.104	TCP	54		443 → 50789 [ACK] Seq=3980 Ack=2428 Win=73728 Len=0
46	3.080196	104.26.8.59	192.168.0.104	TLSv1.3	626		Application Data
47	3.080650	104.26.8.59	192.168.0.104	TLSv1.3	1082		Application Data

Рисунок 3.4 – Проведення ініціації підключення через протокол TLS

No.	Time	Source	Destination	Protocol	Length	Comment	Info
97	3.169966	104.26.8.59	192.168.0.104	TLSv1.3	1445		Application Data
98	3.169966	104.26.8.59	192.168.0.104	TLSv1.3	1110		Application Data, Application Data
100	3.170007	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2816 Ack=15766 Win=131584 Len=0
101	3.171815	104.26.8.59	192.168.0.104	TLSv1.3	506		Application Data
102	3.171815	104.26.8.59	192.168.0.104	TLSv1.3	910		Application Data
103	3.171815	104.26.8.59	192.168.0.104	TLSv1.3	85		Application Data
104	3.171923	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2816 Ack=17105 Win=130048 Len=0
110	3.177183	192.168.0.104	104.26.8.59	TLSv1.3	123		Application Data
112	3.178241	104.26.8.59	192.168.0.104	TLSv1.3	507		Application Data
113	3.178284	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2885 Ack=17558 Win=131584 Len=0
114	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1445		Application Data
115	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
116	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1376		Application Data
117	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
118	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
119	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
120	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1238		Application Data
121	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
122	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
123	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
124	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
125	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
126	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
127	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1320		Application Data, Application Data, Application Data
132	3.179320	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2885 Ack=37321 Win=131584 Len=0
143	3.189276	192.168.0.104	104.26.8.59	TLSv1.3	120		Application Data
148	3.201459	104.26.8.59	192.168.0.104	TLSv1.3	86		Application Data
149	3.201459	104.26.8.59	192.168.0.104	TLSv1.3	85		Application Data

Рисунок 3.5 – Обмін трафіку з корисним навантаження протоколу TLS

Далі, на основі відомостей з підрозділу 1.2, проведено визначення та охарактеризування мережевих ознак ініціації та обміну корисним навантаженням VPN сесії, найбільш популярними VPN протоколами. Спершу, буде розглянуто роботу протоколу OpenVPN, рисунок 3.6, ініціація з’єднання починається з TCP сесії, далі поверх неї ініціюється SSLv2 сесія, перші декілька пакетів якої проводять ініціацію та після рядку 228, починається передача корисного навантаження OpenVPN, при цьому пакети ініціації є досить малого розміру, значно меншого ніж у стандартній TLS сесії, рисунок 3.4 рядок 30, відбувається це тому що, встановлення TLS або її старшої версії SSL, відбувається з передачею значень наборів шифрів, номеру сесії та інших розширень, та коли ці значення не додаються до вмісту, як у випадку з OpenVPN, це може послугувати ознакою виявлення його роботи. Наступною ознакою є його надсилання пакетів підтвердження, ACK, з однаковими інтервалами, що зображено з рядку 240 та нижче, що може стати ознакою його виявлення під час передавання корисного навантаження.

216	26.123974	192.168.0.104	54.36.174.140	TCP	66	64737 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
217	26.139878	54.36.174.140	192.168.0.104	TCP	66	443 → 64737 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
218	26.139995	192.168.0.104	54.36.174.140	TCP	54	64737 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
219	26.141691	192.168.0.104	54.36.174.140	SSLv2	70	
220	26.158792	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=1 Ack=17 Win=64256 Len=0
224	26.636380	54.36.174.140	192.168.0.104	SSLv2	82	
225	26.637556	192.168.0.104	54.36.174.140	SSLv2	368	
226	26.654259	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=29 Ack=331 Win=64128 Len=0
227	26.655009	54.36.174.140	192.168.0.104	SSLv2	78	
228	26.656400	54.36.174.140	192.168.0.104	SSLv2	1514	Encrypted Data, Encrypted Data
229	26.656448	192.168.0.104	54.36.174.140	TCP	54	64737 → 443 [ACK] Seq=331 Ack=1513 Win=131328 Len=0
230	26.673037	54.36.174.140	192.168.0.104	SSLv2	934	
231	26.678250	192.168.0.104	54.36.174.140	SSLv2	82	
232	26.737530	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=2393 Ack=359 Win=64128 Len=0
233	26.737586	192.168.0.104	54.36.174.140	SSLv2	681	
234	26.753837	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=2393 Ack=986 Win=64128 Len=0
235	26.783312	54.36.174.140	192.168.0.104	SSLv2	240	
236	26.784250	192.168.0.104	54.36.174.140	SSLv2	90	
237	26.805443	54.36.174.140	192.168.0.104	SSLv2	305	
238	26.807348	192.168.0.104	54.36.174.140	SSLv2	90	
239	26.826505	54.36.174.140	192.168.0.104	SSLv2	299	
240	26.867003	192.168.0.104	54.36.174.140	TCP	54	64737 → 443 [ACK] Seq=1058 Ack=3075 Win=131328 Len=0
241	26.914719	192.168.0.104	54.36.174.140	SSLv2	90	
242	26.975250	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1094 Win=64128 Len=0
243	26.987316	192.168.0.104	54.36.174.140	SSLv2	145	
244	27.007862	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1185 Win=64128 Len=0
245	27.007971	192.168.0.104	54.36.174.140	SSLv2	232	Encrypted Data
246	27.027490	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1363 Win=64128 Len=0
249	27.047882	192.168.0.104	54.36.174.140	SSLv2	224	
250	27.064905	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1533 Win=64128 Len=0
263	27.247711	192.168.0.104	54.36.174.140	SSLv2	157	
267	27.263848	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1636 Win=64128 Len=0
269	27.263904	192.168.0.104	54.36.174.140	SSLv2	157	
270	27.281554	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1739 Win=64128 Len=0
271	27.281605	192.168.0.104	54.36.174.140	SSLv2	157	
272	27.303106	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1842 Win=64128 Len=0
274	27.306853	192.168.0.104	54.36.174.140	SSLv2	157	
277	27.326157	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=1945 Win=64128 Len=0
279	27.343580	192.168.0.104	54.36.174.140	SSLv2	157	
287	27.359633	54.36.174.140	192.168.0.104	TCP	54	443 → 64737 [ACK] Seq=3075 Ack=2048 Win=64128 Len=0

Рисунок 3.6 – Ініціація сесії та обмін трафіком з корисним навантаженням протоколу OpenVPN

Наступним буде розглянуто протокол Wireguard, використовуючи відомості з підрозділу 1.2. Цей протокол працює виключно через UDP транспортний протокол, рисунок 3.7, ініціація з’єднання відбувається двома пакетами, рядки 63 та 65, та далі відбувається обмін корисним навантаженням. Ключовою ознакою виявлення його роботи у мережі є наявність трьох пустих байтів, нулі, що у пакетах ініціації, що у пакетах обміну корисним навантаженням, рисунок 3.8, вони виділені блакитним та синім кольором.

No.	Time	Source	Destination	Protocol	Length	Comment	Info
→ 63	12.015490	192.168.0.104	34.51.174.233	WireGu…	190		Handshake Initiation, sender=0x91F3A117
← 65	12.068856	34.51.174.233	192.168.0.104	WireGu…	134		Handshake Response, sender=0xC61B740C, receiver=0x91F3A117
66	12.078262	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=0, datalen=80
67	12.078262	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=1, datalen=80
68	12.090028	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=2, datalen=80
69	12.122833	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=3, datalen=80
70	12.133762	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=4, datalen=80
71	12.134224	192.168.0.104	34.51.174.233	WireGu…	170		Transport Data, receiver=0xC61B740C, counter=5, datalen=96
72	12.166562	192.168.0.104	34.51.174.233	WireGu…	122		Transport Data, receiver=0xC61B740C, counter=6, datalen=48
73	12.166701	192.168.0.104	34.51.174.233	WireGu…	138		Transport Data, receiver=0xC61B740C, counter=7, datalen=64
74	12.166756	192.168.0.104	34.51.174.233	WireGu…	218		Transport Data, receiver=0xC61B740C, counter=8, datalen=144
75	12.179275	192.168.0.104	34.51.174.233	WireGu…	186		Transport Data, receiver=0xC61B740C, counter=9, datalen=112
76	12.179765	192.168.0.104	34.51.174.233	WireGu…	202		Transport Data, receiver=0xC61B740C, counter=10, datalen=128
77	12.206451	192.168.0.104	34.51.174.233	WireGu…	138		Transport Data, receiver=0xC61B740C, counter=11, datalen=64
78	12.208609	192.168.0.104	34.51.174.233	WireGu…	138		Transport Data, receiver=0xC61B740C, counter=12, datalen=64
79	12.233529	192.168.0.104	34.51.174.233	WireGu…	138		Transport Data, receiver=0xC61B740C, counter=13, datalen=64
80	12.264393	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=14, datalen=80
81	12.264549	192.168.0.104	34.51.174.233	WireGu…	122		Transport Data, receiver=0xC61B740C, counter=15, datalen=48
82	12.266714	34.51.174.233	192.168.0.104	WireGu…	282		Transport Data, receiver=0x91F3A117, counter=0, datalen=208
83	12.269000	192.168.0.104	34.51.174.233	WireGu…	154		Transport Data, receiver=0xC61B740C, counter=16, datalen=80
84	12.269102	192.168.0.104	34.51.174.233	WireGu…	122		Transport Data, receiver=0xC61B740C, counter=17, datalen=48
86	12.296690	34.51.174.233	192.168.0.104	WireGu…	138		Transport Data, receiver=0x91F3A117, counter=1, datalen=64
87	12.297653	192.168.0.104	34.51.174.233	WireGu…	122		Transport Data, receiver=0xC61B740C, counter=18, datalen=48
88	12.304342	34.51.174.233	192.168.0.104	WireGu…	282		Transport Data, receiver=0x91F3A117, counter=2, datalen=208

Рисунок 3.7 – Ініціація сесії та обмін трафіком з корисним навантаженням протоколу Wireguard

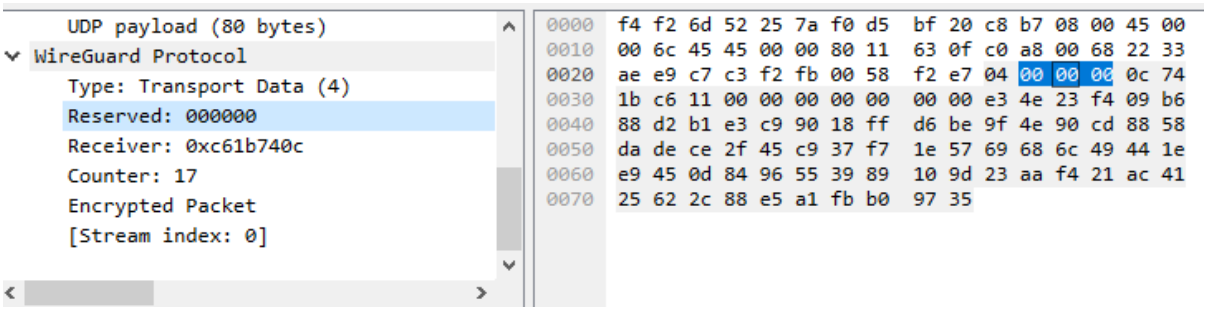


Рисунок 3.8 – Вміст заголовку даних прикладного рівня протоколу Wireguard

На останок, буде проведено аналіз трафіку, що утворює програмний комплекс з обфускації трафіку на основі протоколів VLess та XTLS-Vision. Після початку роботи, клієнт встановлює окрему TCP та TLS сесію з проксі сервером для кожного з'єднання що вже існувало, та власне буде створено. Щоб продемонструвати роботу протоколів обфускації, було застосовано не тільки фільтр для відображення діалогу між проксі та користувачем, а також фільтр на відображення трафіку, що надходить від певного порту користувача до проксі, таким чином, зручно виводячи вигляд стандартної сесії, утвореної програмою, рисунок 3.9. З наведеного рисунку помітно, що ініціація з'єднання відбувається однаковим чином із стандартною TLS сесією, рисунок 3.4, тобто кількість пакетів ініціації сягає близько шести, з яких три відповідають за TCP, рядки номер 1819, 1945, 1946, та ще три за встановлення TLS сесії, рядки номер 2123, 2218, 2647. При цьому, розмір пакетів TLS ініціації є

стандартним, на відміну від OpenVPN. Після чого, починається передача зашифрованого корисного навантаження, у якого інтервали передачі пакетів підтвердження, ACK, є різними, як і в стандартної роботи TLS зв’язку, це рядки номер 2857, 2867, 2871 та інші. Для додаткової наочності роботи програми, було додано ще одне відображення утвореного трафіку, тільки для іншого порту, рисунок 3.10. Описані сесії TCP та TLS, встановлювались між користувачем та проксі сервером у якості тунелю, для передачі даних прикладних протоколів, та відповідно, під корисним навантаженням могли відбуватись будь-які протоколи, що їх ініціювала програма користувача. Тим не менш, як зазначалось у підрозділах 2.1 та 2.4, існує проблема виявлення TLS в TLS, а саме: якщо програма користувача бажає встановити TLS сесію з кінцевим ресурсом, тобто проходячи через TCP TLS сесію проксі сервера, XTLS-Vision забезпечує деактивацію тунельної TLS сесії між користувачем та проксі, залишаючи тільки TCP тунель. Відбувається це тільки, після закінчення TLS рукостискання між програмою користувача та кінцевим ресурсом, власне після чого, відбувається обмін повністю зашифрованими TLS пакетами корисного навантаження цієї сесії. TLS пакети корисного навантаження не мають ідентифікаторів сесій TLS зв’язку до яких вони належать, саме тому, коли у TCP тунелі між користувачем та проксі замість тунельної TLS сесії користувача та проксі, що виконується TLS пакетами корисного навантаження, починає проходити TLS сесія програми користувача та кінцевого ресурсу, пакетами корисного навантаження TLS, цензор не помітить підміни, зміни. Через цю причину, на відображенні Wireshark не можливо продемонструвати роботу XTLS-Vision.

No.	Time	Source	Destination	Protocol	Length	Comment	Info
1819	52.337107	192.168.0.104	64.227.47.212	TCP	66		62567 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1945	52.383055	64.227.47.212	192.168.0.104	TCP	66		443 → 62567 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
1946	52.383208	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
2123	52.516500	192.168.0.104	64.227.47.212	TLSv1.3	571		Client Hello (SNI=brpalukh.space)
2217	52.559531	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=1 Ack=518 Win=64128 Len=0
2218	52.560546	64.227.47.212	192.168.0.104	TLSv1.3	1514		Server Hello, Change Cipher Spec, Application Data
2219	52.560994	64.227.47.212	192.168.0.104	TLSv1.3	995		Application Data, Application Data, Application Data
2220	52.561043	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [ACK] Seq=518 Ack=2402 Win=131328 Len=0
2647	52.858234	192.168.0.104	64.227.47.212	TLSv1.3	118		Change Cipher Spec, Application Data
2781	52.940296	192.168.0.104	64.227.47.212	TLSv1.3	1262		Application Data
2783	52.940472	192.168.0.104	64.227.47.212	TLSv1.3	120		Application Data
2792	52.948813	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=2402 Ack=582 Win=64128 Len=0
2856	52.989721	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=2402 Ack=1790 Win=62976 Len=0
2857	52.990179	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=2402 Ack=1856 Win=62976 Len=0
2866	53.002680	64.227.47.212	192.168.0.104	TLSv1.3	1262		Application Data
2867	53.002680	64.227.47.212	192.168.0.104	TCP	1514		443 → 62567 [ACK] Seq=3610 Ack=1856 Win=62976 Len=1460 [TCP PDU reassembled in 2868]
2868	53.002680	64.227.47.212	192.168.0.104	TLSv1.3	988		Application Data
2869	53.002680	64.227.47.212	192.168.0.104	TLSv1.3	1011		Application Data
2871	53.002762	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [ACK] Seq=1856 Ack=6961 Win=131328 Len=0
3187	53.131218	192.168.0.104	64.227.47.212	TLSv1.3	1363		Application Data
3188	53.131422	192.168.0.104	64.227.47.212	TLSv1.3	1394		Application Data
3195	53.133834	192.168.0.104	64.227.47.212	TLSv1.3	78		Application Data
3197	53.133926	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [FIN, ACK] Seq=4529 Ack=6961 Win=131328 Len=0
3308	53.175160	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=6961 Ack=4505 Win=60416 Len=0
3311	53.177235	64.227.47.212	192.168.0.104	TLSv1.3	1394		Application Data

Рисунок 3.9 – Ініціація сесії та обмін трафіком з корисним навантаженням протоколів обфускації VLess та XTLS-Vision для порту №1

No.	Time	Source	Destination	Protocol	Length	Comment	Info
1956	52.394740	192.168.0.104	64.227.47.212	TCP	66		62576 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2045	52.441116	64.227.47.212	192.168.0.104	TCP	66		443 → 62576 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
2046	52.441233	192.168.0.104	64.227.47.212	TCP	54		62576 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
2144	52.521017	192.168.0.104	64.227.47.212	TLSv1.3	571		Client Hello (SNI=brpalukh.space)
2235	52.566659	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=1 Ack=518 Win=64128 Len=0
2247	52.568769	64.227.47.212	192.168.0.104	TLSv1.3	1514		Server Hello, Change Cipher Spec, Application Data
2248	52.568769	64.227.47.212	192.168.0.104	TLSv1.3	995		Application Data, Application Data, Application Data
2250	52.568982	192.168.0.104	64.227.47.212	TCP	54		62576 → 443 [ACK] Seq=518 Ack=2402 Win=131328 Len=0
2525	52.772131	192.168.0.104	64.227.47.212	TLSv1.3	118		Change Cipher Spec, Application Data
2634	52.856539	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=2402 Ack=582 Win=64128 Len=0
2846	52.981456	192.168.0.104	64.227.47.212	TLSv1.3	1262		Application Data
2847	52.981615	192.168.0.104	64.227.47.212	TLSv1.3	158		Application Data
2904	53.026259	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=2402 Ack=1790 Win=62976 Len=0
2905	53.026691	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=2402 Ack=1894 Win=62976 Len=0
2937	53.038267	64.227.47.212	192.168.0.104	TLSv1.3	1262		Application Data
2939	53.039086	64.227.47.212	192.168.0.104	TCP	1514		443 → 62576 [ACK] Seq=3610 Ack=1894 Win=62976 Len=1460 [TCP PDU reassembled in 2940]
2940	53.039086	64.227.47.212	192.168.0.104	TLSv1.3	988		Application Data
2941	53.039086	64.227.47.212	192.168.0.104	TLSv1.3	872		Application Data
2942	53.039183	192.168.0.104	64.227.47.212	TCP	54		62576 → 443 [ACK] Seq=1894 Ack=6822 Win=131328 Len=0
3233	53.148899	192.168.0.104	64.227.47.212	TLSv1.3	1174		Application Data
3234	53.149134	192.168.0.104	64.227.47.212	TLSv1.3	1444		Application Data
3335	53.195243	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=6822 Ack=4404 Win=62976 Len=0
3353	53.198349	64.227.47.212	192.168.0.104	TLSv1.3	1334		Application Data
3453	53.240295	192.168.0.104	64.227.47.212	TLSv1.3	78		Application Data

Рисунок 3.10 – Ініціація сесії та обмін трафіком з корисним навантаженням протоколів обфускації VLess та XTLS-Vision для порту №2

Далі, буде створено таблицю демонстрації ефективності роботи програмного комплексу з обфускації трафіку, в порівнянні з протоколами VPN та стандартним мережевим трафіком, таблиця 3.1. Для порівняння будуть використовуватись ознаки виявлення не стандартної мережевої активності, зокрема VPN протоколів. Будуть порівнюватись ознак ініціації сесії, обміну пакетами корисного навантаження та не стандартного використання протоколу TLS.

Таблиця 3.1 – Порівняння існуючих аналогів програмного комплексу з обфускації трафіку

Характеристика	Стандартний TLS трафік	Трафік утворений протоколом OpenVPN	Трафік утворений протоколом Wireguard	Трафік утворений власною розробкою з обфускацією
1	2	3	4	5
Ініціація сесії	Звичайна, стандартна	Не звичайна, менша кількість пакетів, помітний протокол	Не звичайна, не притаманна для типу транспорту, помітний протокол	Стандартна TLS ініціація
Обмін корисним навантаженням	Звичайний, стандартний	Має помітні алгоритми підтвердження, ACK	Звичайний для типу транспорту, помітний протокол	Стандартний обмін корисним навантаженням TLS
Відбитки протоколу TLS	Звичайні відбитки	Сильно відрізняються	Відсутні	Підроблені, не відрізняються від звичайних
Виявлення TLS-в-TLS	TLS-в-TLS не проводиться	Помітно проведення TLS-в-TLS	Відсутні	Деактивація одного TLS тунелю при виявлені іншого

Таким чином, програмний комплекс з обфускації трафіку на основі протоколів VLess та XTLS-Vision, забезпечують проведення TLS сесії тунелювання, що ідеально відповідає усім характеристиках звичайного, стандартного TLS з’єднання.

3.4 Висновки до розділу 3

У розділі було реалізовано програмний комплекс з обфускації трафіку, чим власне було створено ефективне рішення, для приховання VPN-з’єднань від засобів виявлення ISP. На стороні проксі сервера було розгорнуто XRay-core з підтримкою протоколу VLess у режимі XTLS-Vision, що забезпечило високий рівень маскування трафіку, завдяки якому з’єднання виглядає як звичайний TLS трафік. Було впроваджено сертифіковане TLS-з’єднання із використанням реального доменного імені, що значно ускладнює ідентифікацію проксі сервера, та впроваджено

механізм fallback з використанням локального nginx-сервера, що дозволяє додатково приховувати ознаки активності, відводячи трафік на умовний вебресурс у разі неправильного UUID.

На клієнтській стороні використано графічний клієнт Nekoray, який дозволяє зручно налаштовувати маршрутизацію трафіку, у тому числі з можливістю прямого підключення до певних IP-адрес без використання проксі. Було впроваджено переадресацію через TUN-інтерфейс, що дозволило обійти обмеження системного проксі, пов’язані з DNS-запитами та несумісністю деяких додатків. Загальний інтерфейс клієнта надає зручний спосіб управління профілями з'єднання та забезпечує візуальний контроль над активними тунелями.

Також, було проведено практичний аналіз трафіку за допомогою Wireshark. Порівняння між звичайним, VPN та обфускованим трафіком підтвердило ефективність реалізованого підходу, трафік, що проходить через V2Ray з XTLS-Vision, не має характерних ознак VPN-протоколів, таких як визначені порти, відомі TLS-розширення чи нетипова поведінка під час встановлення з’єднання. Усі ключові відмінності між класичними VPN-засобами та запропонованим обфускованим тунелем були продемонстровані та проаналізовані у відповідній порівняльній таблиці.

Таким чином, результати, представлені у цьому розділі, доводять, що створений програмний комплекс є повноцінним і дієвим засобом для обходу інтернет-цензури та виявлення VPN, зберігаючи при цьому стабільність з’єднання, швидкість передачі даних та зручність у користуванні.

ВИСНОВОК

У межах даної роботи було проведено комплексне дослідження проблеми виявлення та блокування VPN з’єднань з боку інтернет-провайдерів і систем глибокої інспекції трафіку, а також реалізовано програмний комплекс, який забезпечує ефективну обфускацію інтернет-трафіку з метою приховування ознак використання VPN.

У першому розділі було розглянуто теоретичні засади роботи інтернет-мереж, VPN технологій та методів інспекції трафіку. Особливу увагу приділено механізмам, які використовуються для виявлення VPN з’єднань, таким як, аналіз шаблонів TLS-рукостискань, специфічних портів, часових характеристик та інших. Також описано стратегії обфускації, включно з використанням стеганографії та поліморфізму протоколів. Та було проаналізовано ознаки робіт VPN технологій у мережевому трафіку.

Другий розділ присвячено технічній реалізації обфускації, вибору відповідних інструментів та протоколів, зокрема VLess + XTLS-Vision, а також розробці та опису архітектури власного програмного комплексу. Було обґрунтовано вибір мови програмування Golang та супутніх технологій Qt/C++ для GUI, JSON для створення конфігураційних файлів комплексу та Linux-середовище для серверної частини. Окрему увагу приділено схемам роботи клієнта і сервера, принципам обміну ключами та шифрування, а також тестуванню стабільності з’єднання під час роботи в умовах блокувань. Було розроблено набір технологій, що забезпечить надання VPN послуг користувачеві, та не буде відображати відомих ознак VPN технологій під час своєї роботи.

У третьому розділі було побудовано програмний комплекс, що складається з серверної і клієнтської частини, та забезпечить обфускацію VPN трафіку. Також, проведено експериментальне дослідження ефективності запропонованого рішення. Було здійснено аналіз роботи програмного комплексу в реальному мережевому середовищі з використанням інструментів типу Wireshark, проведено порівняння зі звичайним та VPN-трафіком, та окреслено відповідні відмінності. Отримані результати підтверджують, що використання обфускації на основі VLess + XTLS-

Vision суттєво ускладнює або повністю унеможливлює ідентифікацію VPN-трафіку користувача, зберігаючи при цьому низьку затримку та високу пропускну здатність.

Результати роботи мають практичну цінність, оскільки створений програмний комплекс може бути використаний не лише як інструмент захисту конфіденційності у мережі, а й як детальна інструкція з налаштування серверної та клієнтської частини VPN-сервісу з обфускацією трафіку. Робота структурована таким чином, щоб навіть користувачі без глибоких технічних знань могли повторити описані кроки та забезпечити собі безпечний доступ до інтернету.

Особливої актуальності робота набуває в умовах російської агресії та спроб масового впровадження інтернет-цензури на тимчасово окупованих територіях України. Російські ISP можуть не блокувати виявлений VPN трафік, а тільки фіксувати осіб, що його утворювали, та проводити на основі цього, подальші репресивні дії, тому обгунтування вибору технології обфускації виходило не з фактичного тестування різних технологій обфускації на апараті цензури агресора, а з відритих відомостей, щодо роботи та ознак мережевих протоколів, наукових та аматорських досліджень з цієї теми за останні роки, та з відомостей осіб, що користуються інтернетом під умовами цензури інших країн, із використанням технологій обфускації. Більшість цих осіб перебувають у Китаї, який власне є "лідером" з цензурування інтернету, тому враховуючи особливість росіян копіювати усі світові технології, поки реалізований програмний комплекс з обфускації успішно працює у Китаї, цензори агресора імовірніше всього, не зможуть його виявити.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет. Wikipedia. : вебсайт. URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82 (дата звернення: 02.05.2025).

2. Mohammad Shehab. Evaluating the Effectiveness of StealthProtocols and Proxying in Hiding VPN Usage. *Journal of Computational and Cognitive Engineering*. 2024. С. 186–194 (дата звернення: 02.05.2025).

3. Isara Anantavrasilp. Packet encapsulation. Researchgate : вебсайт. URL: https://www.researchgate.net/figure/Packet-encapsulation-TCP-IP-architecture-encapsulates-the-data-from-the-upper-layer-by_fig4_49288737 (дата звернення: 03.05.2025).

4. Pippin. How Packets Flow. OpenVPN community wiki. 17.12.2019 : вебсайт. URL: https://community.openvpn.net/Pages/HowPacketsFlow (дата звернення: 04.05.2025).

5. Diwen Xue, Reethika Ramesh. OpenVPN is Open to VPN Fingerprinting. *Communications of the ACM*. 2025. С. 79 – 87. ISSN 0001-0782 (дата звернення: 04.05.2025).

6. 4.1 HTTPS Layer Establishment. Learn Microsoft : вебсайт. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-sstp/7e5b2134-b4bf-435a-85bf-bfe0313fd889 (дата звернення: 06.05.2025).

7. Pavel Valach. Analysis and detection of WireGuard traffic : Bachelor’s thesis / Czech Technical University in Prague Faculty of Information Technology. Прага, 2023. 46 с (дата звернення: 06.05.2025).

8. Mingshi Wu, Jackson Sippe. How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic. *USENIX*, Анахайм, CA, США, 9–11 серп. 2023 (дата звернення: 07.05.2025).

9. Roya Ensaf, David Fifield. Examining How the Great Firewall Discovers Hidden Circumvention Servers. *ACM IMC 2015*, Tokyo, Japan, 28-30 жовт. 2015 (дата звернення: 07.05.2025).

10. Sharing a modified Shadowsocks as well as our thoughts on the cat-and-mouse game. Github. 15.10.2022 : вебсайт. URL: https://github.com/net4people/bbs/issues/136 (дата звернення: 08.05.2025).

11. Junwon Lee, Heejo Lee. An SSH predictive model using machine learning with web proxy session logs. *International Journal of Information Security*. 2022. URL: https://doi.org/10.1007/s10207-021-00555-6 (дата звернення: 09.05.2025).

12. VMess. V2RAY : вебсайт. URL: https://www.v2ray.com/en/configuration/protocols/vmess.html (дата звернення: 10.05.2025).

13. VLESS. V2FLY : вебсайт. URL: https://www.v2fly.org/config/protocols/vless.html (дата звернення: 10.05.2025).

14. uTLS. Github : вебсайт. URL: https://github.com/refraction-networking/utls (дата звернення: 12.05.2025).

15. XTLS Vision, fixes TLS in TLS, to the star and beyond. Github. 31.10.2022 : вебсайт. URL: https://github-com.translate.goog/XTLS/Xray-core/discussions/1295?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=ru&_x_tr_pto=wapp (дата звернення: 14.05.2025).

16. Sharing a modified Shadowsocks as well as our thoughts on the cat-and-mouse game. Github. 15.10.2022 : вебсайт. URL: https://github.com/net4people/bbs/issues/136 (дата звернення: 14.05.2025).

17. Outline VPN. Wikipedia : вебсайт. URL: https://en.wikipedia.org/wiki/Outline_VPN (дата звернення: 15.05.2025).

18. OpenConnect. Wikipedia : вебсайт. URL: https://uk.wikipedia.org/wiki/OpenConnect (дата звернення: 16.05.2025).

19. Go (programming language). Wikipedia : вебсайт. URL: https://en.wikipedia.org/wiki/Go_(programming_language) (дата звернення: 16.05.2025).

20. Qt. Wikipedia : вебсайт. URL: https://uk.wikipedia.org/wiki/Qt (дата звернення: 16.05.2025).

21. JSON. Wikipedia : вебсайт. URL: https://uk.wikipedia.org/wiki/JSON (дата звернення: 17.05.2025).

22. Ubuntu. Wikipedia : вебсайт. URL: https://uk.wikipedia.org/wiki/Ubuntu (дата звернення: 17.05.2025).

23 Gralla, Preston. How the Internet works. Que Publishing, 1998. 10–14 с. (дата звернення: 18.05.2025).

24. Edwards, James, Bramante, Richard. Networking self-teaching guide: OSI, TCP/IP, LANs, MANs, WANs, implementation, management, and maintenance. John Wiley \& Sons, 2015. 503–505 с. (дата звернення: 18.05.2025).

25. Alshalan, Abdullah. A survey of mobile VPN technologies. *IEEE Communications Surveys \& Tutorials*. 2015. Т. 18, вип. 2. С. 1177–1196. (дата звернення: 19.05.2025).

26. Olteanu, Vladimir, Niculescu,. Design of SOCKS Version 6. *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*. С. 126—128. (дата звернення: 20.05.2025).

27. Crist, Eric. Mastering OpenVPN. Packt Publishing Ltd, 2015. 185–187 с. (дата звернення: 20.05.2025).

28. Donenfeld, Jason A. WireGuard: Next Generation Kernel Network Tunnel. *NDSS*. 2017. С. 1–12. (дата звернення: 21.05.2025).

29. Ndatinya, Vivens. Network forensics analysis using Wireshark. *International Journal of Security and Networks*. 2015. Т. 10 : 2. С. 91–106. (дата звернення: 21.05.2025).

30. Luo, Jie, Bao. A method of Shadowsocks (R) traffic identification based on protocol analysis. *2021 IEEE 21st International Conference on Communication Technology (ICCT)*. 2021. С. 6–10. (дата звернення: 21.05.2025).

31. Reti, Daniel. Secure (s) hell: Introducing an SSH deception proxy framework. *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment*. 2021. С. 1–6. (дата звернення: 21.05.2025).

32. Wu, Hua. Rt-cbch: Real-time vpn traffic service identification based on sampled data in high-speed networks. *IEEE Transactions on Network and Service Management*. 2023. Т. 21 : 1. С. 88–107 (дата звернення: 22.05.2025).

33. Free the internet. Mullvad : вебсайт. URL: https://mullvad.net/en (дата звернення: 22.05.2025).

34. Як це працює. Outline : вебсайт. URL: https://getoutline.org/uk/ (дата звернення: 22.05.2025).

35. OpenConnect. OpenConnect : вебсайт. URL: https://www.infradead.org/openconnect/ (дата звернення: 22.05.2025).

36. Zhang, Yidan. Network traffic identification of several open source secure proxy protocols. *International Journal of Network Management*. 2021. Т. 31 : 2. (дата звернення: 23.05.2025).

37. Chabbi, Milind. A study of real-world data races in Golang. *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 2022 (дата звернення: 23.05.2025).

38. Peng, Gang. CDN: Content distribution network. *arXiv preprint cs/0411069*. 2004 (дата звернення: 23.05.2025).

39. Higham, PD. QT dispersion. *British heart journal*. 1994. Т. 71 : 6 (дата звернення: 23.05.2025).

40. Petre, Marian. UML in practice. *2013 35th international conference on software engineering (icse)*. 2013. С. 722–731 (дата звернення: 24.05.2025).

41. Plotog, Ioan. VPS technology and applications. *2010 3rd International Symposium on Electrical and Electronics Engineering (ISEEE)*. 2010. С. 351–354 (дата звернення: 24.05.2025).

42. Iqbal, Haroon. Wireshark as a tool for detection of various LAN attacks. *International Journal of Computer Sciences and Engineering*. 2019. С. 833–837 (дата звернення: 24.05.2025).

43. Frankel, Sheila. Guide to IPsec VPNs:.. *US Department of Commerce, Technology Administration, National Institute of*. 2005 (дата звернення: 25.05.2025).

44. Guezzaz, Azidine. A new hybrid network sniffer model based on Pcap language and sockets (Pcapsocks). *International Journal of Advanced Computer Science and Applications*. 2016 (дата звернення: 25.05.2025).

45. Newman, Chris. Using tls with imap, pop3 and acap. *Rfc-editor*. 01.01.1999 : вебсайт. URL: https://www.rfc-editor.org/rfc/rfc2595 (дата звернення: 25.05.2025).

ДОДАТКИ

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
" 20 " березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської кваліфікаційної роботи на тему:

Розробка програмного комплексу приховування vpn активності за допомогою обфускації трафіку
08-72.БКР.017.00.000.ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.ф.-м.н., доцент
_____ Шиян А. А.
" _____ " _____

Вінниця – 2025 р.

1. Найменування та область застосування

Розробка програмного комплексу приховування vpn активності за допомогою обфускації трафіку. Область застосування: обмін інформацією у мережі інтернет.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 97 від 20.03.2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: створення програмного комплексу, що забезпечує встановлення обфускованої VPN сесії.

3.2 Призначення: розроблений комплекс виконує приховання ознак використання VPN протоколів у мережі.

4. Джерела розробки

4.1. Mingshi Wu, Jackson Sippe. How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic. *USENIX*, Анахайм, CA, США, 9–11 серп. 2023.

4.2. Roya Ensaf, David Fifield. Examining How the Great Firewall Discovers Hidden Circumvention Servers. *ACM IMC 2015*, Tokyo, Japan, 28-30 жовт. 2015.

4.3 VLESS. V2FLY : вебсайт. URL: https://www.v2fly.org/config/protocols/vless.html.

4.4. Sharing a modified Shadowsocks as well as our thoughts on the cat-and-mouse game. Github. 15.10.2022 : вебсайт. URL: https://github.com/net4people/bbs/issues/136.

4.5 XTLS Vision, fixes TLS in TLS, to the star and beyond. Github. 31.10.2022 : вебсайт. URL: https://github-com.translate.goog/XTLS/Xray-core/discussions/1295?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=ru&_x_tr_pto=wapp.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний повинен приховувати IP-адресу кінцевого ресурсу, до якого звертається користувач;

5.1.2 Утворений трафік між користувачем та проксі не повинен містити ознак VPN технології;

5.1.3 Розгортання комплексу повинне бути доступним для осіб, з різним рівнем обізнаності в мережевих технологіях.

5.2 Вимоги до надійності:

5.2.1 Наявність процедур аутентифікації, забезпечення конфіденційності та цілісності з’єднання;

5.2.2 Проведення логування на стороні сервера та клієнта, для фіксування та ідентифікації помилок, якщо такі виникнуть;

5.2.3 Програмний комплекс повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i3–8100 / AMD Ryzen 3 1200 і подібні до них;

– оперативна пам’ять – не менше 2048 Mb;

– середовище функціонування – серверна частина потребує ОС на базі Linux, клієнтська частина, підтримується мультиплатформно;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.1 та 3.2.

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити відповідний рівень шифрування даних під час їх передачі у мережі.

7.2 Неможливість отримання доступу до проксі сервера, у випадку не пройденої аутентифікації.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Початок	Закінчення
1.	Визначення теми бакалаврської роботи	20.03.2025	23.03.2025
2.	Аналіз предметної області обраної теми	24.03.2025	13.04.2025
3.	Розробка алгоритму роботи	14.04.2025	27.04.2025
4.	Написання бакалаврської роботи	28.04.2025	25.05.2025
5.	Передзахист бакалаврської кваліфікаційної роботи	26.05.2025	27.05.2025
6.	Виконання правок та виправлень бакалаврської кваліфікаційної роботи	28.05.2025	02.06.2025
7.	Захист бакалаврської кваліфікаційної роботи	09.06.2025	10.06.2025

10. Порядок контролю та прийому

10.1 До приймання бакалаврської кваліфікаційної роботи надається:

– ПЗ до бакалаврської кваліфікаційної роботи;

- демонстрація результату бакалаврської кваліфікаційної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв _____ Палюх М. Є.

Додаток Б. Лістинг коду файлу конфігурації V2Ray серверної частини комплексу

```
{
 "log": {
  "loglevel": "warning"
 },
 "routing": {
  "rules": [],
  "domainStrategy": "AsIs"
 },
 "inbounds": [
  {
   "port": 443,
   "protocol": "vless",
   "settings": {
    "clients": [
     {
      "id": "188fc747-0a10-42d1-8501-870194e015db",
      "flow": "xtls-rprx-vision"
     }
    ],
    "decryption": "none",
    "fallbacks": [
     {
      "dest": "8080",
      "xver": 1
     }
    ]
   },
   "streamSettings": {
    "network": "tcp",
    "security": "tls",
    "tlsSettings": {
     "rejectUnknownSni": true,
     "minVersion": "1.2",
     "alpn": [
      "http/1.1",
      "h2"
     ],
     "certificates": [
      {
       "ocspStapling": 3600,
       "certificateFile": "/etc/letsencrypt/live/brpalukh.space/fullchain.pem",
       "keyFile": "/etc/letsencrypt/live/brpalukh.space/privkey.pem"
      }
     ]
    }
   },
   "sniffing": {
    "enabled": true,
    "destOverride": [
     "http",
     "tls"
    ]
   }
  }
```

```
 ],
 "outbounds": [
  {
    "protocol": "freedom",
    "tag": "direct"
  }
 ],
 "policy": {
     "levels": {
        "0": {
           "handshake": 2, // The handshake time limit when the connection is established, in seconds, the default value is 4, it is recommended to be different from the default value
           "connIdle": 120 // Connection idle time limit in seconds, the default value is 300, it is recommended to be different from the default value
        }
     }
  }
}

server {
	listen 127.0.0.1:8002 default_server;
	listen [::1]:8002 default_server;

	# SSL configuration
	#
	# listen 443 ssl default_server;
	# listen [::]:443 ssl default_server;
	#
	# Note: You should disable gzip for SSL traffic.
	# See: https://bugs.debian.org/773332
	#
	# Read up on ssl_ciphers to ensure a secure configuration.
	# See: https://bugs.debian.org/765782
	#
	# Self signed certs generated by the ssl-cert package
	# Don't use them in a production server!
	#
	# include snippets/snakeoil.conf;

	root /var/www/html;

	# Add index.php to the list if you are using PHP
	index index.html index.htm index.nginx-debian.html;

	server_name _;

	location / {
		# First attempt to serve request as file, then
		# as directory, then fall back to displaying a 404.
		try_files $uri $uri/ =404;
		#proxy_pass http://ftp.debian.org/debian/
		#auth_basic "Administrator's Area";
		#auth_basic_user_file /etc/htpasswd;
	}

	# pass PHP scripts to FastCGI server
	#
	#location ~ \.php$ {
	#	include snippets/fastcgi-php.conf;
```

```
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
}
```

Додаток В. Ілюстративний матеріал

Розробка програмного комплексу приховування vpn активності за допомогою обфускації трафіку

Виконав студент групи 1КІТС-21б Палюх М. Є.
Науковий керівник к.ф.-м.н., доцент каф. МБІС Шиян А. А.

Актуальність:

Актуальність роботи полягає у тому, що існують ситуації, при яких користувачу необхідно приховати IP-адресу кінцевого ресурсу, до якого він бажає звернутись, для цього він може почати використовувати технологій проксування або VPN. При цьому, може існувати наступна потреба у прихованні факту використання VPN активності.

Мета:

Метою роботи є розробка програмного комплексу для обфускації VPN-трафіку, який забезпечує його маскування під звичайні TLS-з'єднання.

Завдання:

- Дослідити роботу VPN протоколів.
- Проаналізувати сучасні протоколи обфускації.
- Реалізувати клієнтську і серверну частини програмного забезпечення.
- Провести тестування та обгрунтувати доцільність використання технологій VLESS + XTLS-Vision.

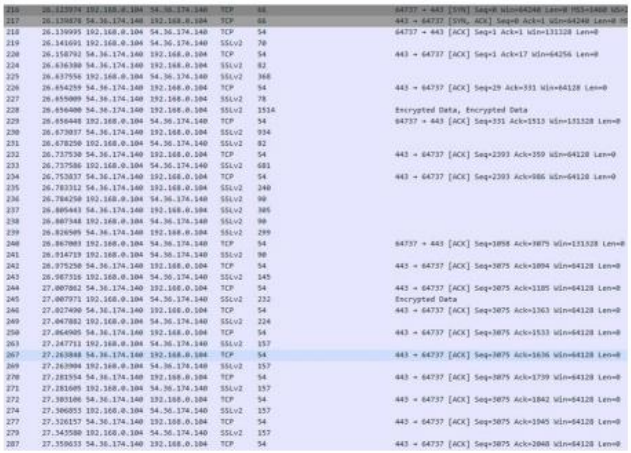
Кожен користувач інтернету під'єднуються до нього через свого провайдера ISP, який у свою чергу, також під'єднується до вищого за рівнем провайдера. Власне ці два ISP, Tier 3 та Tier 2,можуть контролюватися однією централізованою державною інфраструктурою, яка діє в межах території перебування користувача. Таким чином, місцевий держ. режим може запровадити цензуру у користуванні інтернетом для користувачів.

Цензура російських провайдерів може блокувати звернення до певного набору IP-адрес для їхніх користувачів, або/та певний тип трафіку у їх мережах. Крім цього, російські провайдери, окрім блокування, можуть проводити фіксування даних користувачів, хто таку діяльність проводив, без відому останніх. Зокрема небезпеці піддаються користувачі на окупованих територіях України.

Ініціація сесії та обмін її пакетами VPN протоколів мають свої семантичні та статистичні ознаки, що може виявляти російський цензор. Далі будуть наведені такі ознаки у роботі деяких VPN протоколів.

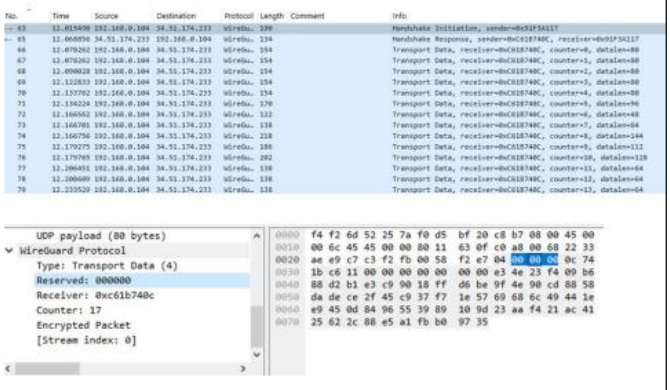
Протокол OpenVPN

Проводить ініціацію з'єднання схожу на TLS, проте старішою версією та не відповідного розміру пакетами. Статистичною ознакою на надсилання пакетів підтвердження ACK, з однаковими інтервалами.



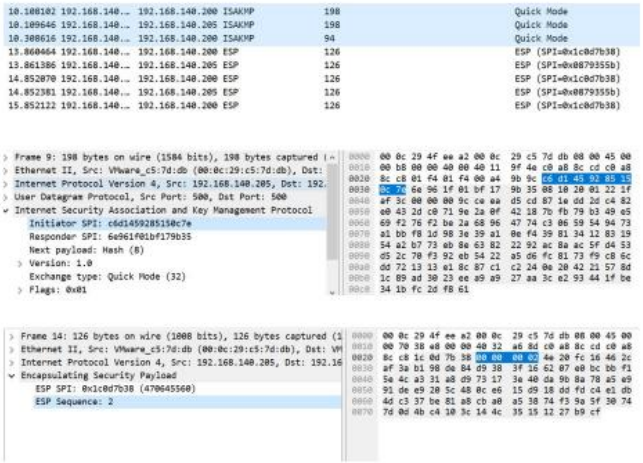
Протокол Wireguard

Цей протокол працює виключно через UDP транспортний протокол, ініціація з'єднання відбувається двома пакетами, та далі відбувається обмін корисним навантаженням. Ключовою ознакою виявлення його роботи у мережі є наявність трьох пустих байтів, нулі, що у пакетах ініціації, що у пакетах обміну корисним навантаженням.



Протокол IPSec

Кожен заголовок прикладного рівня IPSec починається з значення SPI користувача, яке він не змінює у ході спілкування. Під час встановлення сесії та обміну даними, заголовки мають фіксовану структуру, як наприклад, в обміні даними, після SNI 4 байти використовуються для позначення порядку пакета.



Деякі протоколи можуть підтверджувати свою присутність у з'єднанні, у результаті проведення активного зондування, наприклад протокол MS SSTP, роботу якого можна виявити, якщо виконати HTTP запит до сервера до якого звертається користувач, методом SSTP_DUPLEX_POST та з URI адресою /sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/, в результаті чого, сервер відповість підтвердженням наявності MS SSTP.

Рішенням для продовження використання технологій VPN та не бути поміченим цензором є проведення обфускації такого трафіку. Обфускація походить від англійського слова "obfuscation" – заплутування, та має два основних підходи до реалізації – стеганографічні та поліморфні. Мета стеганографічного підходу це зробити обфускований трафік схожим на дозволений трафік, а мета поліморфізму це зробити трафік обходу не схожим на заборонений трафік..

Проте не всі існуючі технології обфускації є досконалими, та мають свої недоліки.

Наприклад ShadowSocks, працює шляхом утворення програмою користувача Socks запиту, який потім шифрується, та напряму передається у TCP тунель між користувачем та його проксі сервером. Та оскільки, виглядає він як зашифрована цифрова маса, яку передає TCP, виявити його вдається за відношенням 1 до 0, його корисного навантаження.

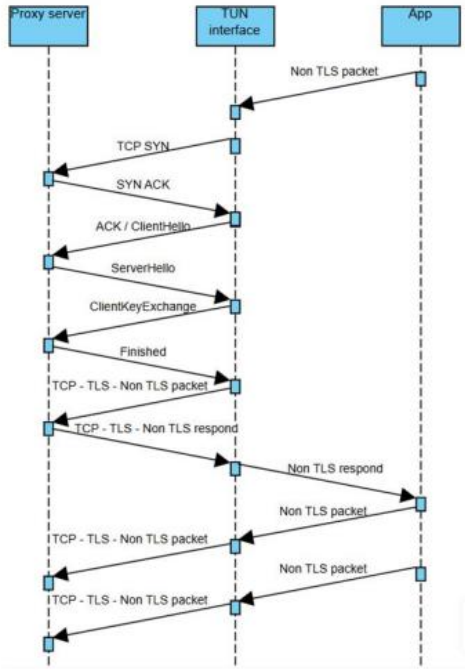
Використання тунелю SSH, як способу обфускації, теж має свої недоліки. Зазвичай для пакетів консольного керування, для чого найчастіше використовується SSH, пакет не перебільшує розміру в 100 байтів. Та трафік керування консоллю, повинен бути рідким та мати малий об'єм пакетів.

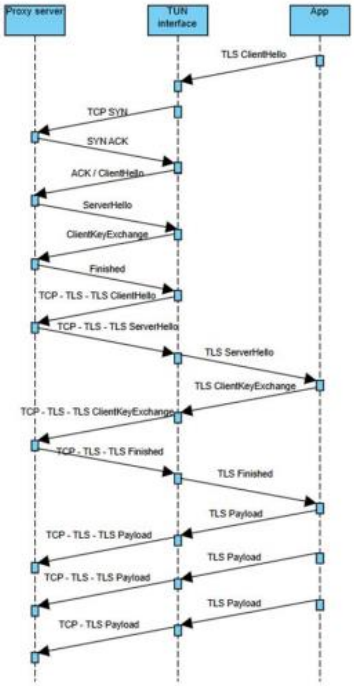
Протокол VMess, отримуючи пакети програм користувача, не проводить їх паддинг та має певну структуру заголовків прикладного рівня. А також, вразливий перед активним зондуванням.

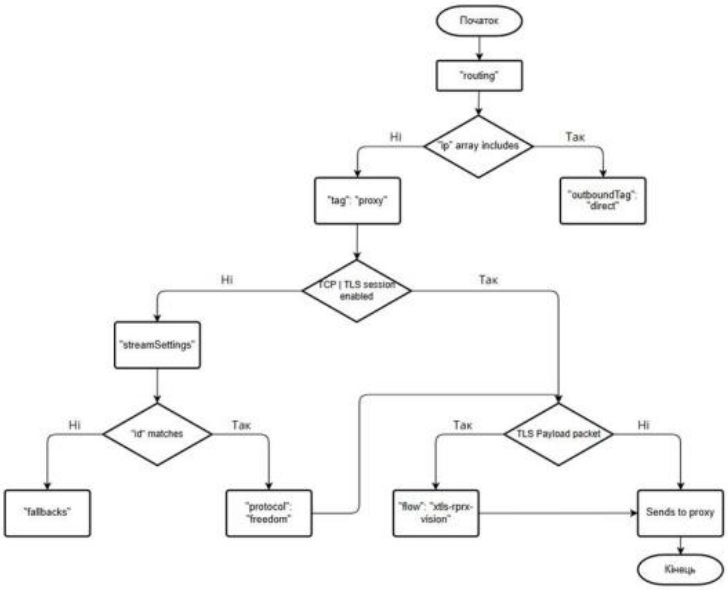
На основі аналізу та порівнянь різних технологій обфукації, для виконання роботи було обрано зв'язку VLess та XTLS-Vision.

VLess, отримуючи дані від програми користувача, проводить їх паддинг додавання значення UUID, для аутентифікації користувача, протокол не проводить шифрування даних, тому ніколи не працює без транспорту прикладного рівня.

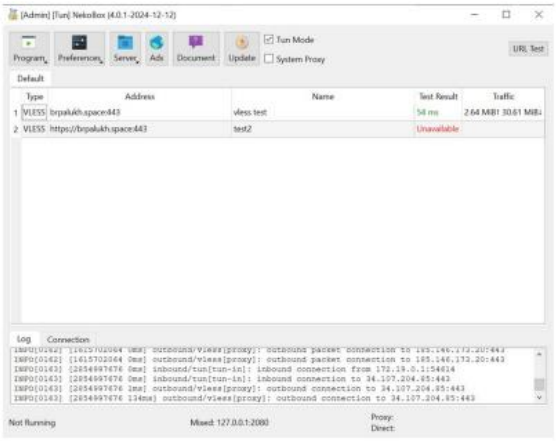
Протокол TLS, є найчастішим вибором для транспортування прикладного рівня у багатьох веб служб, тому у якості стеганографічного підходу обфускації, він є одним з найкращих кандидатів. Тим не менш, використання TLS потребує вирішення деяких його особливостей.

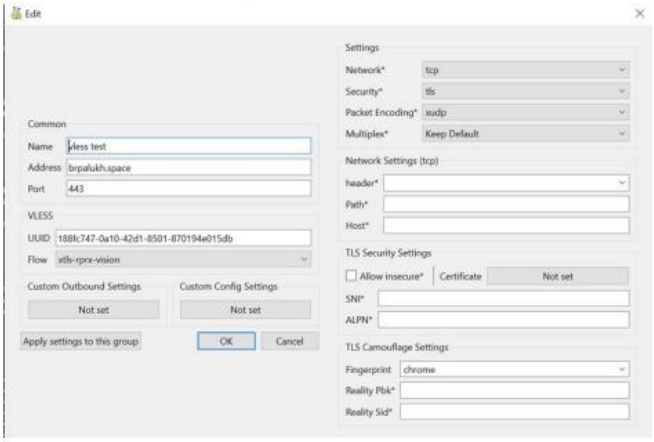






```
"inbounds": [
  {
    "port": 443,
    "protocol": "vless",
    "settings": {
      "clients": [
        {
          "id": "188fc747-0a10-42d1-8501-870194e015db",
          "flow": "xtls-rprx-vision"
        }
      ],
      "decryption": "none",
      "fallbacks": [
        {
          "dest": "8080",
          "xver": 1
```





Ознаки стандартного мережевого підключення TLS, без технологій VPN та обфускації є наступними.
Після встановлення сесії, всі дані передаються зашифровано, пакетами Application Data, маючи різний розмір, packet lengths. При цьому, до передачі корисного навантаження відбувається обмін, в середньому шістьма пакетами, після чого, трафік передається зі зворотнім надсиланням пакетів підтвердження, ACK, з різними інтервалами часу.

No.	Time	Source	Destination	Protocol	Length	Comment	Info
26	2.834071	192.168.0.104	104.26.8.59	TCP	66		50789 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
27	2.839245	104.26.8.59	192.168.0.104	TCP	66		443 → 50789 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM WS=8192
28	2.839317	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0
29	2.839917	192.168.0.104	104.26.8.59	TCP	1454		50789 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=1400 [TCP PDU reassembled in 30]
30	2.839917	192.168.0.104	104.26.8.59	TLSv1.3	408		Client Hello (SNI=www.myip.com)
31	2.856401	104.26.8.59	192.168.0.104	TCP	54		443 → 50789 [ACK] Seq=1 Ack=1755 Win=73728 Len=0
33	2.872592	104.26.8.59	192.168.0.104	TLSv1.3	1514		Server Hello, Change Cipher Spec
34	2.872787	104.26.8.59	192.168.0.104	TCP	1514		443 → 50789 [ACK] Seq=1461 Ack=1755 Win=73728 Len=1460 [TCP PDU reassembled in 35]
35	2.872787	104.26.8.59	192.168.0.104	TLSv1.3	592		Application Data

No.	Time	Source	Destination	Protocol	Length	Comment	Info
97	3.169966	104.26.8.59	192.168.0.104	TLSv1.3	1445		Application Data
98	3.169966	104.26.8.59	192.168.0.104	TLSv1.3	1110		Application Data, Application Data
100	3.170007	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2816 Ack=15766 Win=131584 Len=0
101	3.171815	104.26.8.59	192.168.0.104	TLSv1.3	506		Application Data
102	3.171815	104.26.8.59	192.168.0.104	TLSv1.3	910		Application Data
103	3.171815	104.26.8.59	192.168.0.104	TLSv1.3	85		Application Data
104	3.171923	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2816 Ack=17105 Win=130048 Len=0
110	3.177183	192.168.0.104	104.26.8.59	TLSv1.3	123		Application Data
112	3.178241	104.26.8.59	192.168.0.104	TLSv1.3	507		Application Data
113	3.178284	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2885 Ack=17558 Win=131584 Len=0
114	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1445		Application Data
115	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
116	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1376		Application Data
117	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
118	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
119	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
120	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1238		Application Data
121	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
122	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
123	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
124	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
125	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
126	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1514		Application Data
127	3.179151	104.26.8.59	192.168.0.104	TLSv1.3	1320		Application Data, Application Data, Application Data
132	3.179320	192.168.0.104	104.26.8.59	TCP	54		50789 → 443 [ACK] Seq=2885 Ack=37321 Win=131584 Len=0
143	3.189276	192.168.0.104	104.26.8.59	TLSv1.3	120		Application Data
148	3.201459	104.26.8.59	192.168.0.104	TLSv1.3	86		Application Data
149	3.201459	104.26.8.59	192.168.0.104	TLSv1.3	85		Application Data

Ознаки роботи комплексу з обфускації, у разі ініціації нової сесії наступні.

Коли клієнт встановлює з'єднання, він створює звичайну TCP та TLS сесію з проксі-сервером — точно так само, як це відбувається під час доступу до звичайного HTTPS-сайту. Кількість і розміри пакетів ініціації при цьому відповідають стандартному TLS з'єднанню, 6 пакетів — 3 TCP + 3 TLS. Після завершення TLS-рукостискання, клієнт передає зашифровані дані — але з різною частотою ACK-пакетів, як у звичайному інтернет-з'єднанні. XTLS-Vision дозволяє програмі користувача напряму встановити TLS-з'єднання з сайтом, минуючи TLS-тунель між клієнтом і проксі, після завершення рукостискання.

No.	Time	Source	Destination	Protocol	Length	Comment	Info
1819	52.337107	192.168.0.104	64.227.47.212	TCP	66		62567 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1945	52.383055	64.227.47.212	192.168.0.104	TCP	66		443 → 62567 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
1946	52.383208	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
2123	52.516500	192.168.0.104	64.227.47.212	TLSv1.3	571		Client Hello (SNI=brpalukh.space)
2217	52.559531	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=1 Ack=518 Win=64128 Len=0
2218	52.560546	64.227.47.212	192.168.0.104	TLSv1.3	1514		Server Hello, Change Cipher Spec, Application Data
2219	52.560994	64.227.47.212	192.168.0.104	TLSv1.3	995		Application Data, Application Data, Application Data
2220	52.561043	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [ACK] Seq=518 Ack=2402 Win=131328 Len=0
2647	52.858234	192.168.0.104	64.227.47.212	TLSv1.3	118		Change Cipher Spec, Application Data
2781	52.940296	192.168.0.104	64.227.47.212	TLSv1.3	1262		Application Data
2783	52.940472	192.168.0.104	64.227.47.212	TLSv1.3	120		Application Data
2792	52.948813	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=2402 Ack=582 Win=64128 Len=0
2856	52.989721	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=2402 Ack=1790 Win=62976 Len=0
2857	52.990179	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=2402 Ack=1856 Win=62976 Len=0
2866	53.002680	64.227.47.212	192.168.0.104	TLSv1.3	1262		Application Data
2867	53.002680	64.227.47.212	192.168.0.104	TCP	1514		443 → 62567 [ACK] Seq=3610 Ack=1856 Win=62976 Len=1460 [TCP PDU reassembled in 2868]
2868	53.002680	64.227.47.212	192.168.0.104	TLSv1.3	988		Application Data
2869	53.002680	64.227.47.212	192.168.0.104	TLSv1.3	1011		Application Data
2871	53.002762	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [ACK] Seq=1856 Ack=6961 Win=131328 Len=0
3187	53.131218	192.168.0.104	64.227.47.212	TLSv1.3	1363		Application Data
3188	53.131422	192.168.0.104	64.227.47.212	TLSv1.3	1394		Application Data
3195	53.133834	192.168.0.104	64.227.47.212	TLSv1.3	78		Application Data
3197	53.133926	192.168.0.104	64.227.47.212	TCP	54		62567 → 443 [FIN, ACK] Seq=4529 Ack=6961 Win=131328 Len=0
3308	53.175160	64.227.47.212	192.168.0.104	TCP	54		443 → 62567 [ACK] Seq=6961 Ack=4505 Win=60416 Len=0
3311	53.177235	64.227.47.212	192.168.0.104	TLSv1.3	1394		Application Data

No.	Time	Source	Destination	Protocol	Length	Comment	Info
1956	52.394740	192.168.0.104	64.227.47.212	TCP	66		62576 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2045	52.441116	64.227.47.212	192.168.0.104	TCP	66		443 → 62576 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
2046	52.441233	192.168.0.104	64.227.47.212	TCP	54		62576 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
2144	52.521017	192.168.0.104	64.227.47.212	TLSv1.3	571		Client Hello (SNI=brpalukh.space)
2235	52.566659	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=1 Ack=518 Win=64128 Len=0
2247	52.568769	64.227.47.212	192.168.0.104	TLSv1.3	1514		Server Hello, Change Cipher Spec, Application Data
2248	52.568769	64.227.47.212	192.168.0.104	TLSv1.3	995		Application Data, Application Data, Application Data
2250	52.568982	192.168.0.104	64.227.47.212	TCP	54		62576 → 443 [ACK] Seq=518 Ack=2402 Win=131328 Len=0
2525	52.772131	192.168.0.104	64.227.47.212	TLSv1.3	118		Change Cipher Spec, Application Data
2634	52.856539	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=2402 Ack=582 Win=64128 Len=0
2846	52.981456	192.168.0.104	64.227.47.212	TLSv1.3	1262		Application Data
2847	52.981615	192.168.0.104	64.227.47.212	TLSv1.3	158		Application Data
2904	53.026259	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=2402 Ack=1790 Win=62976 Len=0
2905	53.026691	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=2402 Ack=1894 Win=62976 Len=0
2937	53.038267	64.227.47.212	192.168.0.104	TLSv1.3	1262		Application Data
2939	53.039086	64.227.47.212	192.168.0.104	TCP	1514		443 → 62576 [ACK] Seq=3610 Ack=1894 Win=62976 Len=1460 [TCP PDU reassembled in 2940]
2940	53.039086	64.227.47.212	192.168.0.104	TLSv1.3	988		Application Data
2941	53.039086	64.227.47.212	192.168.0.104	TLSv1.3	872		Application Data
2942	53.039183	192.168.0.104	64.227.47.212	TCP	54		62576 → 443 [ACK] Seq=1894 Ack=6822 Win=131328 Len=0
3233	53.148899	192.168.0.104	64.227.47.212	TLSv1.3	1174		Application Data
3234	53.149134	192.168.0.104	64.227.47.212	TLSv1.3	1444		Application Data
3335	53.195243	64.227.47.212	192.168.0.104	TCP	54		443 → 62576 [ACK] Seq=6822 Ack=4404 Win=62976 Len=0
3353	53.198349	64.227.47.212	192.168.0.104	TLSv1.3	1334		Application Data
3453	53.240295	192.168.0.104	64.227.47.212	TLSv1.3	78		Application Data

Характеристика	Стандартний TLS трафік	Трафік утворений протоколом OpenVPN	Трафік утворений протоколом Wireguard	Трафік утворений власною розробкою з обфускацією
1	2	3	4	5
Ініціація сесії	Звичайна, стандартна	Не звичайна, менша кількість пакетів, помітний протокол	Не звичайна, не притаманна для типу транспорту, помітний протокол	Стандартна TLS ініціація
Обмін корисним навантаженням	Звичайний, стандартний	Має помітні алгоритми підтвердження, ACK	Звичайний для типу транспорту, помітний протокол	Стандартний обмін корисним навантаженням TLS
Відбитки протоколу TLS	Звичайні відбитки	Сильно відрізняються	Відсутні	Підроблені, не відрізняються від звичайних
Виявлення TLS-в-TLS	TLS-в-TLS не проводиться	Помітно проведення TLS-в-TLS	Відсутні	Деактивація одного TLS тунелю при виявлені іншого

Результати роботи

У роботі розроблено та налаштовано програмний комплекс для обфускації VPN-трафіку, який дозволяє маскувати VPN з’єднання під звичайний TLS/HTTPS трафік. Проведено аналіз існуючих рішень і обгрунтовано вибір протоколу VLESS + XTLS-Vision як найбільш ефективного для обфускації. Результати роботи мають практичну цінність, оскільки створений програмний комплекс може бути використаний не лише як інструмент захисту конфіденційності у мережі, а й як детальна інструкція з налаштування серверної та клієнтської частини VPN-сервісу з обфускацією трафіку. Робота структурована таким чином, щоб навіть користувачі без глибоких технічних знань могли повторити описані кроки та забезпечити собі безпечний доступ до інтернету. Особливої актуальності робота набуває в умовах російської окупації та спроб масового впровадження інтернет-цензури на тимчасово окупованих територіях України.

Дякую за увагу

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Розробка програмного комплексу приховування vpn активності за допомогою обфукації трафіку

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. 1КІТС-21б

Керівник доцент Шиян А.А. _____

Показники звіту подібності

Strike Plagiarism	
Оригінальність	97,2 %
Загальна схожість	2,80 %

Аналіз звіту подібності (відмітити потрібне)

- ☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Палюх М.Є.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

- ☐ **Допустити до захисту**

_____
_____
_____
_____

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)