Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії ім. В. В. Поповського
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

ГЮІК.ХХХХХХ.003ПЗ
(позначення документа)

Аналіз програм, стійких до блокування ресурсів в системах Deep Packet Inspection

Analysis of Programs Resistant to Resource Blocking in Deep Packet Inspection Systems

(тема)

Виконав: студент 6 курсу, групи ІКІм-19-1
напряму
172. Телекомунікації та радіотехніка
(код і повна назва напряму)
Кодаченко Р. Ю.
(прізвище, ініціали)

Керівник доц. Токар Л. О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ О. В. Лемешко
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій

Кафедра Інфокомунікаційної інженерії ім. В.В. Поповського

Рівень вищої освіти другий (магістерський)

Спеціальність 172 Телекомунікації та радіотехніка

Спеціалізація Інфокомунікаційна інженерія
(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Кодаченку Руслану Юрійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи: «Аналіз програм, стійких до блокування ресурсів в системах Deep Packet Inspection», затверджена наказом по університету від « 20 » жовтня 2020 р. № 1396
2. Термін подання студентом роботи до екзаменаційної комісії 22.12.2020р.
3. Вихідні дані до роботи:
1. Технологія глибокої інспекції пакетів (DPI).
2. Середнє значення пропускної здатності 45,6 Мбіт/с при підключенні по SSH-тунелю, кількість втрачених сегментів — 5 258.
3. Середнє значення пропускної здатності 39,3 Мбіт/с при підключенні по OpenConnect-тунелю в режимі TLS, кількість втрачених сегментів — 87 062.
4. Середнє значення пропускної здатності 40,3 Мбіт/с при підключенні по OpenConnect-тунелю в режимі DTLS, кількість втрачених сегментів — 88 703.
5. Середнє значення пропускної здатності 38,9 Мбіт/с із застосуванням обфускації OpenVPN, кількість втрачених сегментів — 343 311.
6. Середнє значення пропускної здатності 45,1 Мбіт/с із застосуванням програми stunnel, кількість втрачених сегментів — 267 906.
7. Середнє значення пропускної здатності 28,2 Мбіт/с при підключенні по тунелю wstunnel, кількість втрачених сегментів — 2 678.
4. Перелік питань, що потрібно опрацювати в роботі:
1. Опис та аналіз досліджуваної технології
2. Опис методики проведення дослідження та аналізу програм, стійких до глибокого аналізу пакетів
3. Аналіз та порівняння отриманих в результаті досліду даних
4. Розробка програми для приховання використання факту обходу
5. Порівняння розробленої програми з існуючими рішеннями

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів): презентація на слайдах

1. Вступ
2. Тема та мета
3. Технологія Deep Packet Inspection
4. Приклад застосування технології DPI у компанії
5. Рішення, стійкі до блокування ресурсів системами DPI
6. Методика дослідження програм, стійких до DPI-аналізу
7. Пряме підключення (порт 5201)
8. SSH-тунелювання (порт 22)
9. VPN-SSL. Підключення через OpenConnect в режимі TLS (порт 443)
10. OpenVPN (порт 1194)
11. Обфускація OpenVPN. Застосування stunnel
12. Обфускація OpenVPN. Застосування obfs4
13. Порівняння значень параметрів з'єднання методів обходу DPI
14. Активне зондування
15. Програма із застосування Port Knocking
16. Програма із застосуванням WebSocket
17. Тунель wstunnel
18. Висновки

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доцент Токар Л.О.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Опис та аналіз досліджуваної технології	02.10.2020р.	Виконано
2	Опис методики проведення дослідження та аналізу програм, стійких до глибокого аналізу пакетів	08.10.2020р.	Виконано
3	Аналіз та порівняння отриманих в результаті досліду даних	10.10.2020р	Виконано
4	Розробка програми для приховання використання факту обходу	02.11.2020р.	Виконано
5	Порівняння розробленої програми з існуючими рішеннями	14.11.2020р.	Виконано

Дата видачі завдання 01 09 2020р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н., доц. Токар Л.О
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 71 с., 40 рис., 1 табл., 4 додатка, 44 джерела.

ГЛИБОКА ПЕРЕВІРКА ПАКЕТІВ, ПРОГРАМИ ОБХОДУ, ТУНЕЛЬ, ЦЕНЗУРА, БЛОКУВАННЯ, ВІРТУАЛЬНА ПРИВАТНА МЕРЕЖА.

Об’єкт дослідження – програми, стійкі до блокування ресурсів системами глибокої перевірки мережевих пакетів.

Предмет дослідження – методи обходу блокування інтернет-ресурсів.

Мета роботи – аналіз програм, стійких до блокування ресурсів в системах глибокої перевірки мережевих пакетів та розробка програми для приховання факту використання методу обходу.

Методи дослідження – емпіричний аналіз, формалізація, порівняння та розробка програми для приховання факту використання програми обходу.

Еволюція рівнів перевірки заголовків мережевих пакетів спричинена по різним причиним, зокрема і для контролю користувачів Інтернет, що є загрозою для принципу мережевого нейтралітету.

У роботі виконаний аналіз різних рівнів перевірки заголовків мережевих пакетів. Розглянуто їхні достоїнства та недоліки. Особлива увага приділена дослідженню програм, стійких до глибокого аналізу мережевих пакетів.

Розроблено програму, що здатна приховувати факт використання програми обходу.

Практичне значення роботи полягає у розкритті потенційних програм для обходу систем глибокого аналізу пакетів, що застосовуються в цілях, що порушують мережевий нейтралітет.

ABSTRACT

The report contains: 71 p., 40 fig., 1 table, 4 applications, 44 sources.

DEEP PACKET INSPECTION, BYPASS PROGRAMS, TUNNEL, CENSORSHIP, BLOCKING, VIRTUAL PRIVATE NETWORK.

The research object is programs resistant to resource blocking by deep packet inspection systems.

The subject of research is methods of bypassing blocking by means of deep packet inspection.

An aim of work is an analysis of programs resistant to resource blocking in deep packet inspection systems and program development for hiding the fact of using a bypass method.

Methods of research are empirical analysis, formalization, comparison and program development for hiding the fact of using the bypass program.

The evolution of packet inspection levels is caused by various reasons, including control of Internet users, which is a threat to the net neutrality.

The analysis of different packet inspection levels is performed. Their dignities and disadvantages are considered. The special attention is spared investigating of programs that are resistant to deep packet inspection.

A program is developed that can hide a fact of using the bypass program.

The practical significance of the work is to reveal potential programs for bypassing deep packet inspection systems used for purposes that violate the net neutrality.

ЗМІСТ

Перелік скорочень, умовних позначень, символів, одиниць і термінів 7
Вступ 9
1 Існуючі рівні перевірки заголовків мережевих пакетів 10
1.1 Поверхнева перевірка заголовків мережевих пакетів 11
1.2 Середній рівень перевірки заголовків мережевих пакетів 12
1.3 Глибока перевірка мережевих пакетів 13
1.3.1 Технічні можливості системи DPI 14
1.3.2 Застосування систем DPI як економічного важеля 16
1.3.3 Значення систем DPI для політичного контролю 18
1.4 Висновки до першого розділу 19
2 Аналіз наявних методів, стійких до активного зондування 20
2.1 Активне зондування як допоміжний засіб DPI-системи 20
2.2 Метод загального обходу, заснований на HTTPS 22
2.3 Система обходу Snowflake 24
2.4 Висновки до другого розділу 28
3 Аналіз та дослідження відомих програм, стійких до DPI 29
3.1 Методика дослідження програм, стійких до DPI-аналізу 29
3.2 SSH-тунелювання для приховання трафіку користувача 31
3.3 Virtual Private Network над Secure Sockets Layer 36
3.4 Обфускація трафіку додатків користувача 39
3.4.1 Програма для обфускації трафіку на базі Pluggable Transports ... 41
3.4.2 Програма забезпечення зміни коду вихідної програми 50
3.5 Висновки до третього розділу 54
4 Розробка програми, стійкої до активного зондування 57
4.1 Програма із застосування технології Port Knocking 57
4.2 Програма із застосування протоколу WebSocket 60
4.3 Висновки до четвертого розділу 65
Висновки 66
Перелік джерел посилання 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система
Мбіт/с – мегабіт в секунду
ПЗ – програмне забезпечення
CDN – content delivery network
DNS – domain name system
DPC – deep packet capture
DPI – deep packet inspection
DTLS – datagram transport layer security
GFW – great firewall of China
HMAC – hash-based message authentication code
HTTP – hyper text transfer protocol
HTTPS – hyper text transfer protocol secure
IANA – Internet assigned numbers authority
IP – internet protocol
ISP – internet service provider
MitM – man in the middle
MPI – middle packet inpection
NAT – network address translation
PT – pluggable transports
OSI – open systems interconnections
SMTP – simple mail transfer protocol
SNI – server name identification
SPI – shallow packet inspection
SSH – secure shell
SSL – secure socket layer
TLS – transport layer security
UDP – user datagram protocol
UDS – Unix domain socket
URL – uniform resource locator
VoiP – voice over internet protocol
VPN – virtual private network

VPS – virtual private server

WebRTC – web real-time communications

ВСТУП

Під час стрімкого піднесення Інтернет існував один фундаментальний принцип: до усіх користувачів та вмісту ставляться однакові умови [1, 2]. Транспортна складова Інтернету не піклувалася про те, яка саме передається інформація. Принцип недискримінації, або «мережевий нейтралітет», дозволяв користувачам вільно користуватися Всесвітньою мережею. Недискримінація у різних формах є основою законодавства та політики у сфері зв'язку, що існує протягом десятиліть [3].

Спочатку технологічно недоцільно для постачальників послуг було перевіряти повідомлення та оцінювати їх вміст у режимі реального часу [4]. Але нещодавно виробники телекомунікаційного обладнання розробили так звану технологію глибокої інспекції пакетів (DPI), здатну відстежувати інтернет-зв’язок у режимі реального часу, контролювати вміст та вирішувати, які повідомлення чи програми будуть оброблені швидше [5]. Технологія DPI дозволяє зчитувати поле даних, що дозволяє операторам мережі ідентифікувати та контролювати на точному рівні повсякденне використання Інтернету. Оператори можуть маркувати пакети для швидкісної або повільної обробки, або блокувати пакети на основі того, що вони містять, або який додаток їх надіслав.

Насправді, неправильне використання або зловживання DPI може змінити Інтернет таким, яким ми його знаємо: перетворивши відкриту та інноваційну платформу в форму платних засобів масової інформації [6]. Хоча раннє використання DPI в режимі реального часу інтернет-провайдерами було спрямоване на поширення цільової реклами та зменшення перевантажень, зараз виробники цієї технології програмують її на суцільний контроль інтернет-користувачів [7]. Коли мережевий провайдер встановлює обладнання DPI, це забезпечує його можливістю контролю та монетизації Інтернету способами, які загрожують знищити відкритий характер Інтернету.

Метою атестаційної роботи є дослідження програм, стійких до блокування ресурсів DPI-системою, та порівняння їх між собою, а також розробка програми, що може виступати в якості рішення для приховання факту використання методу обходу DPI-аналізу від більш розвинутих DPI-систем, та її порівняння з вже існуючими рішеннями (програмами).

1 ІСНУЮЧІ РІВНІ ПЕРЕВІРКИ ЗАГОЛОВКІВ МЕРЕЖЕВИХ ПАКЕТІВ

Хоча системи раннього аналізу пакетів просто досліджували інформацію, отриману із заголовків пакетів даних, тепер вони перевіряють як заголовок, так і корисне навантаження [8]. На більш точному рівні інформація, що використовується для маршрутизації пакетів, походить від фізичного, канального, мережевого та транспортного рівня пакета. Корисне навантаження або вміст пакету включає інформацію про те, яка програма передає дані, чи сам вміст пакету зашифрований, і який точний вміст пакета (наприклад, фактичний текст електронного листа). Більш конкретно, під корисним навантаженням можна розуміти дані на рівнях сеансу, презентації та додатків пакета. Ці деталізовані розділи заголовка та корисного навантаження походять від моделі Open Systems Interconnect (OSI) (рис. 1), яка складається з семи рівнів.

Дані	7 прикланий	Доступ до мережевих служб
	6 представлень	Представлення і кодування даних
	5 сеансовий	Управління сеансом зв'язку
Сегменти	4 транспортний	Прямий зв'язок між кінцевими пунктами і надійність
Пакети	3 мережевий	Визначення маршруту і логічна адресація
Кадри	2 канальний	Фізична адресація
Біти	1 фізичний	Робота з середовищем передачі, сигналами і двійковими даними

Рисунок 1.1 – Рівні в моделі OSI

В залежності від аналізу пакету відносно рівнів моделі OSI рівні перевірки пакетів поділяються на поверхневу, середнього рівня та глибоку.

1.1 Поверхнева перевірка заголовків мережевих пакетів

Технології поверхневої інспекції пакетів (SPI) залежать від (відносно) спрощених брандмауерів. Коли сервер надсилає пакет на клієнтський комп'ютер, технології SPI перевіряють інформацію заголовка пакета та аналізують його за чорним списком. У деяких випадках ці брандмауери постачаються із заздалегідь визначеним набором правил, що складають чорний список, за яким оцінюються дані, тоді як в інших адміністратори мережі відповідають за створення та оновлення набору правил. Зокрема, ці брандмауери фокусуються на адресі інтернет-протоколу (IP) джерела та призначення, до якої намагається отримати доступ пакет, та адресі порту пакета.

Якщо інформація заголовка пакета (IP-адреса, номер порту, або їх комбінація [9]) знаходиться в чорному списку, тоді пакет не доставляється. Коли технологія SPI відмовляється доставити пакет, вона, як правило, не повідомляє джерело про те, що пакет відхилено [9].

На рис. 1.2 зображено приклад роботи SPI в корпоративній мережі.

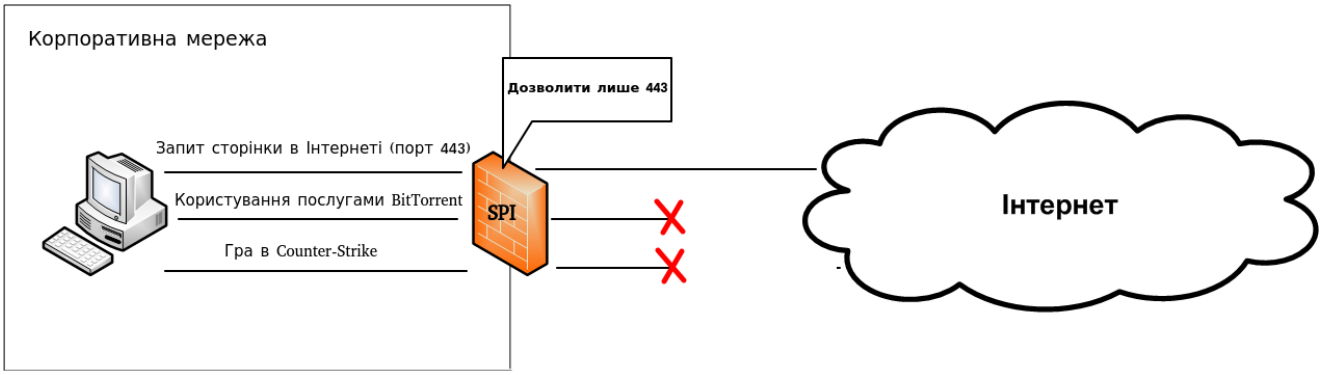

Рисунок 1.2 – Приклад роботи SPI

Більш просунуті форми SPI фіксують журнали вхідної та вихідної інформації про джерело та/або призначення, щоб системний адміністратор пізніше міг переглянути сукупну інформацію заголовка для коригування або створення наборів правил чорного списку.

SPI не може обробляти інформацію, що міститься в заголовках вище четвертого рівня моделі OSI. Так не можна аналізувати заголовки рівнів сеансу, представлень або додатку; він не може «зазирнути» в корисний набір пакета та досліджувати вміст.

1.2 Середній рівень перевірки заголовків мережевих пакетів

У випадку пристроїв середнього рівня перевірки (MPI) передбачено перевірку заголовків пакетів і невелику кількість корисного навантаження. Що важливо, списки розбору, що застосовуються в MPI, є тонкішими у застосуванні, ніж чорні списки. Тоді як останні встановлюють, що допустимо або недопустимо, список розбору дозволяє пропустити або заборонити певні типи пакетів на основі їх типів формату даних та пов'язаного з ними розташування в Інтернеті, а не лише на основі їх IP-адреси.

Таким чином, MPI являє собою еволюцію технологій аналізу пакетів, оскільки він може всебічніше «зчитувати» пакет і вживати більш широкий спектр дій проти пакетів, що потрапляють до їх списків розбору.

Використовуючи пристрої MPI, адміністратори мережі можуть заборонити клієнтським комп’ютерам отримувати флеш-файли з YouTube або файли зображень із сайтів соціальних мереж. Технології MPI можуть надавати пріоритет одним пакетам над іншими, перевіряючи заголовки програм (сигнатури), які знаходяться на рівнях представлення та додатку [10]. З огляду на їх (обмежений) аналіз прикладного рівня пакету, ці пристрої також можуть бути налаштовані на розрізнення нормальних подань протоколу, наприклад, протоколу передачі гіпертексту (HTTP). Вони також можуть детальніше аналізувати пакет та ідентифікувати команди, які пов'язані з протоколом програми, і дозволяти або забороняти з'єднання даних залежно від того, чи є комбінація команд або програм у списку синтаксичного аналізу. Враховуючи статус пристроїв MPI як проксі-серверів додатків, вони також оброблюють характеристики повної інформації в журналах про пакети, на відміну від просто заголовкової інформації, і при інтеграції в ланцюжок довіри можуть дешифрувати трафік даних, перевіряти його, повторно шифрувати трафік і переадресувати його до пункту призначення.

Пристрої MPI мають погану масштабованість: кожна команда програми або протокол, що перевіряється, вимагає унікального шлюзу програми, і перевірка кожного пакета зменшує швидкість, з якою пакети можуть бути доставлені одержувачам [10]. З огляду на ці слабкі місця, пристрої MPI важко розгорнути у великих мережевих операціях, де слід контролювати велику кількість програм. Це обмежує їх користь для постачальників послуг Інтернету, де десятки тисяч програм можуть передавати пакети в будь-який момент.

Незважаючи на те, що пристрої MPI мають обмеження, вони виступають ключовою стороною в технологічних розробках для глибокої перевірки пакетів. Зокрема, їх здатність читати вище четвертого рівня моделі OSI виступає як точка переходу для зчитування всього корисного навантаження. Як результат, ця технологія перевірки є основою на шляху до сучасних технологій глибокої перевірки пакетів.

1.3 Глибока перевірка мережевих пакетів

Обладнання глибокої інспекції пакетів (DPI), як правило, представляє собою дорогі пристрої маршрутизації, які встановлені у великих мережевих концентраторах. Обладнання дозволяє операторам мережі точно визначати походження та вміст кожного пакету даних, що проходить через ці концентратори. Технологія DPI широко використовується для моніторингу та класифікації даних безпосередньо з потоку трафіку. Перевірка пакетів даних на рівнях 3-7 (рис. 2) дозволяє надавати важливу інформацію для операцій та систем підтримки бізнесу, без шкоди для інших служб. У той час як пристрої MPI мають дуже обмежену обізнаність про вміст, пристрої DPI можуть потенційно «зазирнути» всередину всього пакету [11]. Хоча пристрої MPI мають проблеми з масштабуванням, пристрої DPI призначені для визначення того, які програми генерують пакети в режимі реального часу для сотень тисяч транзакцій щосекунди. Вони розроблені для масштабування у великих мережевих середовищах і реагують відразу.

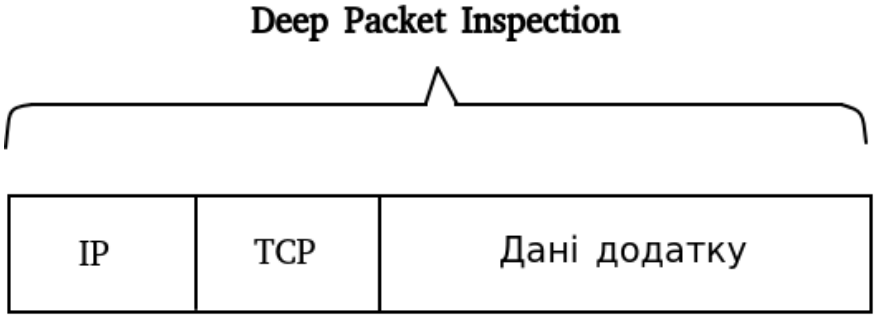

Рисунок 1.3 – Перевірка пакетів технологією DPI

За замовчуванням обладнання DPI аналізує пакет та його характеристики повністю на основі попередньо визначеного набору правил. Такі перевірки

передбачають розгляд рівнів OSI 3-7 для вивчення заголовків пакетів та корисного навантаження для пошуку ознак невідповідності протоколу, шкідливого коду, спаму та будь-яких заздалегідь визначених типів даних, за якими власник мережі хоче стежити або до яких вжити певні заходи. Обладнання визначає та класифікує пакети на основі бази даних підписів. Підписи розробляються шляхом вилучення характерних елементів пакетів, які пов'язані із програмами, що представляють інтерес.

Після ідентифікації пакета, його можна перенаправити, позначити, заблокувати або скинути, обмежити швидкість або повідомити адміністратору мережі. Можна побачити конкретні підписи пакетів, перенаправляючи їх до певного місця в мережі. Можливо, весь трафік простого протоколу передачі пошти (STMP) переадресувати на спеціалізоване обладнання, яке оцінює, чи є трафік спамом чи ні, а згодом відправляє електронне повідомлення до місця призначення. Пакети також можна маркувати, щоб присвоїти їм рівень якості обслуговування; Пакети, чутливі до високих рівнів затримки, можуть отримувати вищий пріоритет для маршрутизації до місця призначення, ніж пакети, на які менше впливає затримка.

Пристрої глибокої перевірки пакетів призначені для досягнення цілого ряду цілей; вони розгортаються з метою безпеки, управління мережею, ідентифікації вмісту та модифікації даних. Цілий ряд соціально-політичних можливостей інтегрований у дизайн технології та відповідає за управління її характеристиками та технічними можливостями. З цією метою пропонується розглянути технічні, економічні та політичні способи використання технології.

1.3.1 Технічні можливості системи DPI

Спочатку DPI мав запропонувати мережевим провайдерам вдосконалені механізми виявлення та запобігання вторгненню, які можуть розпізнавати сучасні загрози та реагувати на них. Для реагування на загрози, що виникають, шаблони DPI можна переконфігурувати та масштабувати для моніторингу великих обсягів трафіку, а також забезпечити реєстрацію та виявлення аномалій. Ведення запису встановлює модель відомої поведінки і дозволяє системі (і системному адміністратору, якщо вони ведуть журнали) перевіряти трафік «поза мережею». Офлайн-аналіз сприяє детальному аналізу трафіку, оскільки він не повинен відбуватися в режимі реального часу, таким чином пом'якшуючи деякі технічні проблеми, пов'язані з поглибленим аналізом пакетів даних, зберігаючи високі

швидкості передачі даних.

Також може використовуватися навчання на основі ведення журналу для розробки очікуваних шаблонів для окремих користувачів та додатків, а також повідомляти адміністраторам, якщо виявляються відхилення.

Цифрові мережі передають все більшу кількість даних, і ключові моменти в мережі вимагають регулярного оновлення, щоб йти в ногу з моделями зростання. Обладнання для глибокої перевірки пакетів призначене для обмеження незручностей, пов’язаних із перевантаженням маршрутизатора. Виявляючи та визначаючи пріоритети пакетів у режимі реального часу, пристрої DPI можуть забезпечити швидшу обробку чутливих до затримки пакетів. В іншому випадку, якщо оператор мережі визначив конкретні програми або типи програм, які суттєво сприяють перевантаженню маршрутизатора, тоді можуть бути встановлені певні правила, щоб обмежити величину пропускної здатності маршрутизатора, яку вони можуть споживати.

На рис 1.4 приведений приклад застосування технології DPI у компанії, що має доступ до мережі Інтернет.

Рисунок 1.4 – Приклад застосування технології DPI у компанії

У деяких випадках обладнання DPI не може відразу ідентифікувати програму, яка створила пакет. Коли це відбувається, оператори мережі часто можуть використовувати технологію глибокого захоплення пакетів (DPC) для збору пакетів в оперативній або внутрішній пам'яті пристрою. Вона дозволяє мережевим адміністраторам проводити аналітичний аналіз пакетів, щоб визначити справжні причини мережевих несправностей, виявити загрози безпеці та забезпечити відповідність передачі даних та використання мережі викладеним політикам.

Використовуючи цю технологію, можна захопити, а потім проаналізувати та ідентифікувати пакетний потік нової програми обміну файлами, який був незнайомий пристрою DPI. Після ідентифікації потоку пакетів цієї нової програми до кожного його пакету могли б застосовуватися набори правил, які б відповідали мережевій політиці провайдера.

Щоб зробити цей останній процес дещо зрозумілішим, звернемось до прикладу. Пакети Skype не можуть бути індентифіковані, через маскування даних заголовків пакетів та шифрування даних корисного навантаження. З огляду на те, що самі пакети повністю зашифровані, а інформація, що міститься в заголовках, є неправдивою, провайдери повинні застосувати інший метод виявлення трафіку Skype. Як рішення, пристрої DPI повинні стежити за конкретним обміном даними, який відбувається, коли користувачі Skype ініціюють голосовий чат. Кожного разу, коли ви контактуєте з кимось, хто використовує Skype, на перший погляд випадковий початковий пакет пакетних обмінів дотримується загальної схеми, яку можна евристично ідентифікувати та співвіднести із програмою Skype [12]. Після того, як додаток буде ідентифіковано, можна перешкоджати або визначати пріоритети для пакетів, що генеруються цією програмою.

У сукупності очевидно, що обладнання DPI забезпечує мережевих адміністраторів інструментами для кращого захисту своїх мереж, реалізації вимог доступу та підвищення якості обслуговування деяких програм.

1.3.2 Застосування систем DPI як економічного важеля

Можливість аналізувати та реагувати на пакети даних у режимі реального часу надає нові можливості для доходу як для інтернет-провайдерів, так і для розробників. Зокрема, ISP може пропонувати різні плани обслуговування, які конкурують на основі того, які програми клієнти можуть використовувати для підключення до Інтернету, пріоритету надання пакетів програм на

маршрутизаторах або швидкості доступу користувачів до веб-сайтів. Інтернет-провайдери можуть також надавати пріоритет власним послугам із «доданою вартістю», таким як протоколу передачі голосу поверх інтернет-протоколу (VoIP), електронній пошті або системі домашньої безпеки, над послугами, що пропонуються їх конкурентами. Власники авторських прав теж можуть спробувати обмежити спільний доступ до файлів, які порушують авторські права, а рекламодавці можуть відстежувати трафік даних для виявлення уподобань споживачів та згодом модифікувати пакети для розміщення цільової реклами.

Компанія «Grand View Research» провела дослідження ринку технології DPI, де зазначила поступове збільшення рівня інвестицій (рис. 1.5).

Рисунок 1.5 – Зростання рівня інвестицій у технології DPI у Європі, Ближньому Сході та Африці

DPI дозволяє провайдерам надалі корегувати свої пропозиції, вибірково дозволяючи програмам підключатися до Інтернету; підключення веб-браузера та поштового клієнта може бути включено до „базового» пакету Інтернету, тоді як додатки для відеоігор або потокової музики можуть бути включені до пакету „преміум». Встановлення DPI та глибокої інтеграції із серверами управління політикою надають переваги перед попередніми мережевими технологіями,

такими як MPI, оскільки один і той самий пристрій може краще опосередковувати різні типи даних. DPI може бути використаний для перевірки трафіку даних та з'ясування того, чи намагаються «зареєстровані» чи «незареєстровані» пристрої отримати доступ до Інтернету.

На рис. 1.6 в якості прикладу наведено, як один з португальських інтернет-провайдерів продає додаткові щомісячні пакети для послуг передачі певних мобільних додатків на основі обмеженого пакету.

Рисунок 1.6 – Модель ціноутворення на основі додатків в Інтернеті

DPI також може слугувати інструментом захисту авторських прав. У такому випадку обмеження порушення авторських прав може позиціонуватися як забезпечення безпеки користувачів, а також забезпечення належного використання мережі, контролюючи дохід, який може бути зменшений через поведінку, що порушує авторські права.

1.3.3 Значення систем DPI для політичного контролю

Встановивши маршрутизатори DPI у ключових точках мереж інтернет-провайдерів, можна дистанційно контролювати комунікації тих, кого підозрюють у незаконних діях, роблячи копії всього трафіку даних або спеціально націлюючи

на один тип трафіку (наприклад, VoIP, веб-перегляд , або однорангова мережа), а не реєстрацію або моніторинг трафіку, що не входить до зазначеного набору правил.

Важливо визнати, що хоча з одного боку, використання DPI може розглядатися як логічна технологія для полегшення державного нагляду, цей режим моніторингу відрізняється від традиційних можливостей прослуховування через широту комунікацій, що відбуваються в Інтернеті. У той час як традиційне прослуховування передає голосовий зв'язок, спостереження за допомогою DPI може фіксувати та проводити аналіз будь-якого типу цифрових транзакцій, будь то голосовий зв'язок, текстовий чат, сеанс веб-перегляду або будь-який інший вид зашифрованої передачі. Таким чином, «прослуховування» на базі DPI, безперечно, розширює підтримку державного нагляду [13].

Оскільки приватні власники авторських прав можуть бути заохочені контролювати за порушеннями файлів, що передаються через цифрові мережі з цивільних причин, уряд може бути зацікавлений у відстеженні та запобіганні передачі вмісту, який він визнав незаконним.

1.4 Висновки до першого розділу

В даному розділі розглянуто рівні перевірки заголовків мережевих пакетів, такі як SPI, MPI та DPI, та вказано на їх роль у застосуванні в інформаційних мережах. Були приведені приклади застосування DPI-технології в економічний та політичній сферах, та вказано на недоліки та переваги її перед іншими рівнями перевірками заголовків мережевих пакетів, що дало змогу виявити потенційну спроможність технології до порушення принципів мережевого нейтралітету, і таким чином, провести додаткове дослідження методів обходу DPI-аналізу.

2 АНАЛІЗ НАЯВНИХ МЕТОДІВ, СТІЙКИХ ДО АКТИВНОГО ЗОНДУВАННЯ

2.1 Активне зондування як допоміжний засіб DPI-системи

В 2010 році Великий брандмауер Китаю (GFW) впровадив інновацію в ідентифікації проксі-серверів: так зване, активне зондування (active probing) [14]. Під час активного зондування цензор видає себе законним клієнтом, встановлюючи власні з’єднання з підозрюваними адресами, щоб побачити, чи не використовуються проксі-сервери. Будь-які адреси, визнані проксі, додаються до чорного списку, щоб у майбутньому доступ до них був заблокований [15]. Для вводу до підсистеми активного зондування слугує набір підозрюваних адрес, що надходять із пасивного спостереження за вмістом з'єднань клієнта. Цензор бачить, що клієнт підключається до пункту призначення, і шляхом перевірки вмісту намагається визначити, який протокол використовується. Коли класифікатор вмісту цензора не може визначити напевно, чи є протокол проксі-протоколом, він передає адресу призначення до активної підсистеми зондування. Потім активний зонд перевіряє, підключившись до пункту призначення, чи насправді це проксі. Рис. 2.1 ілюструє процес активного зондування.

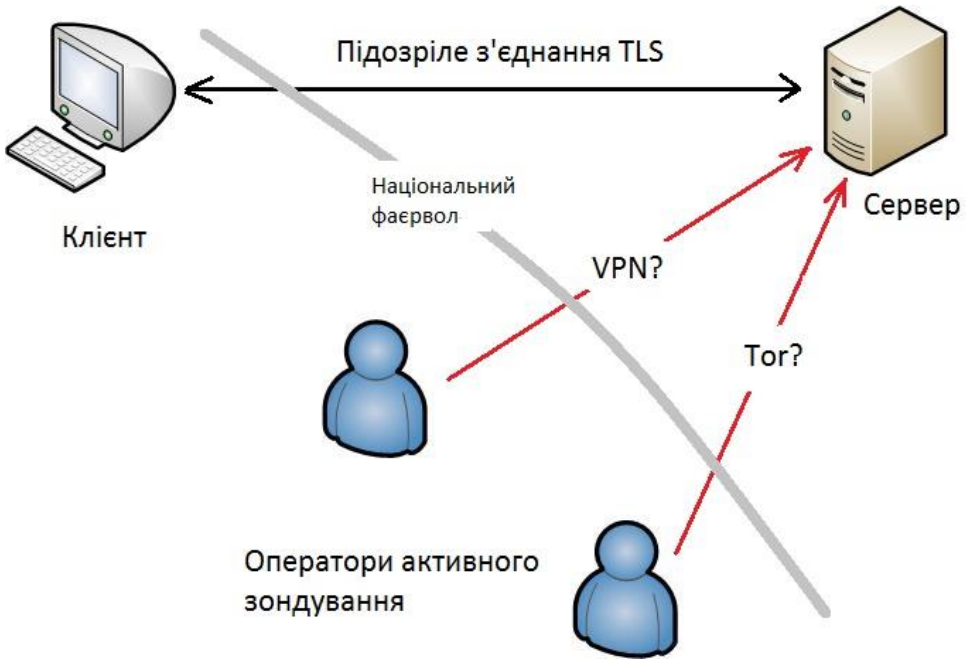

Рисунок 2.1 – Активне зондування

Активне зондування має сенс для цензора, головним обмеженням якого є ризик хибнопозитивні класифікації, що призводять до побічної шкоди. Будь-який класифікатор, заснований виключно на пасивній перевірці вмісту, повинен бути дуже точним (мати низький рівень помилкових спрацьовувань). Активне зондування підвищує точність, блокуючи лише ті сервери, які через активну перевірку визначені справді проксі-серверами. Цензор може уникнути посереднього класифікатора вмісту – він може привести множину набору фактичних проксі-з'єднань, і активні зонди відсіяють його помилкові спрацьовування. Подальша перевага активного зондування, з точки зору цензора, полягає в тому, що воно може працювати асинхронно, окремо від інших обов’язків брандмауера, які вимагають малого часу відгуку.

Термін «активне зондування», що використовується в даній роботі, відрізняється від інших типів активних сканувань тим, що реагує, керуючись спостереженням за з’єднаннями клієнта. Воно відрізняється від активного широкомасштабного сканування портів, при якому цензор регулярно сканує ймовірні порти в Інтернеті, щоб знайти проксі-сервери, незалежно від діяльності клієнта.

GFW є єдиним цензором, який, як відомо, активно використовує зондування. З часом він вдосконалювався, додаючи підтримку нових протоколів та зменшуючи затримку між підключенням клієнта та відправленням зондів. GFW має здатність визначати звичайний протокол Tor [16], певні протоколи для створення віртуальних приватних мереж (VPN) [16] та проксі-сервери, що включають підтримку протоколу захисту транспортного рівня (TLS) через HTTP (HTTPS) [16].

Процес активного зондування починається лише через кілька секунд або хвилин після підключення законним клієнтом. З’єднання зондування надходять із великого діапазону вихідних IP-адрес [16].

Активна система зондування приймає підозрілі адреси як вхідні дані та видає адреси, які потрібно заблокувати, як вихідні дані. Але саме класифікація, що базується на вмісті, створює перелік підозрюваних адрес. Існування активного зондування в певному сенсі було добрим знаком для недавніх методів обходу: обфускація трафіку стала кращою. Якби цензор міг легко ідентифікувати використання протоколів обходу шляхом простого пасивного контролю, то це не призвело б до зайвих клопотів з активним зондуванням.

Сучасні системи обходу DPI повинні бути спроектовані для протистояння

активним зондуючим атакам. Стратегія систем обфускації, таких як, ScrambleSuit, obfs4 та Shadow-socks полягає в аутентифікації клієнтів за допомогою пароля проксі-сервера або відкритого ключа; тобто вимагати додаткової секретної інформації, окрім лише IP-адреси та номера порту. Фронтинг домену бореться з активним зондуванням шляхом спільного розміщення проксі-серверів з важливими веб-сервісами: цензор може бути впевненим, що відбувається обхід, але не може заблокувати проксі без неприпустимих побічних збитків. У «Snowflake» проксі-сервери представляють собою веб-браузери, на яких запущені звичайні однорангові протоколи, автентифіковані за допомогою спільного секрету для кожного з'єднання. Навіть якщо цензор виявляє одного із проксі «Snowflake», він не може перевірити, представляє він проксі-сервер «Snowflake» або щось інше, без попереднього узгодження спільного секрету через механізм брокера «Snowflake».

Відомі методи боротьби з активним зондуванням будуть детальніше розглянуті далі.

2.2 Метод загального обходу, заснований на HTTPS

Метод загального обходу, заснований на HTTPS, відомий як фронтинг доменів, маскує справжнє місце призначення повідомлень клієнта, перенаправляючи їх через загальний веб-сервер або мережу доставки вмісту, на якій розміщено багато веб-сайтів. З точки зору цензора, повідомлення, надходять не до свого фактичного (імовірно заблокованого) пункту призначення, а до якогось іншого переднього домену, блокування якого може призвести до великої супутньої шкоди. Оскільки (з певними застереженнями) цензор не може розрізнити запити HTTPS, керовані доменом, від звичайних запитів HTTPS, він не може блокувати обхід, не блокуючи також передній домен. Фронтинг домену в першу чергу вирішує проблему виявлення за адресою, але також стосується виявлення за вмістом та активного зондування. Сьогодні фронтинг доменів є важливою складовою багатьох систем обходу.

Основною ідеєю фронтингу доменів є використання різних доменних імен на різних рівнях протоколів. Коли ви робите запит HTTPS, доменне ім’я сервера, до якого ви намагаєтесь отримати доступ, зазвичай відображається в трьох місцях, які видимі цензору:

- запит системи доменних імен (DNS);

- розширення індикації імені сервера TLS (SNI) клієнта;
- сертифікат TLS сервера;
- і в одному місці, яке не видно цензору, оскільки воно зашифровано: заголовок хосту HTTP.

У звичайному запиті одне й те саме доменне ім’я з’являється у всіх чотирьох місцях, і всі вони, крім заголовка «Host», надають цензору легку основу для блокування. Різниця в запиті, керованому фронтовим доменом, полягає в тому, що доменне ім’я в заголовку хосту, що знаходиться всередині запиту, не збігається з доменом, який з’являється в інших місцях, зовні. На рис. 2.2 показано перші кроки клієнта, який робить запит, спрямований на фронтовий домен.

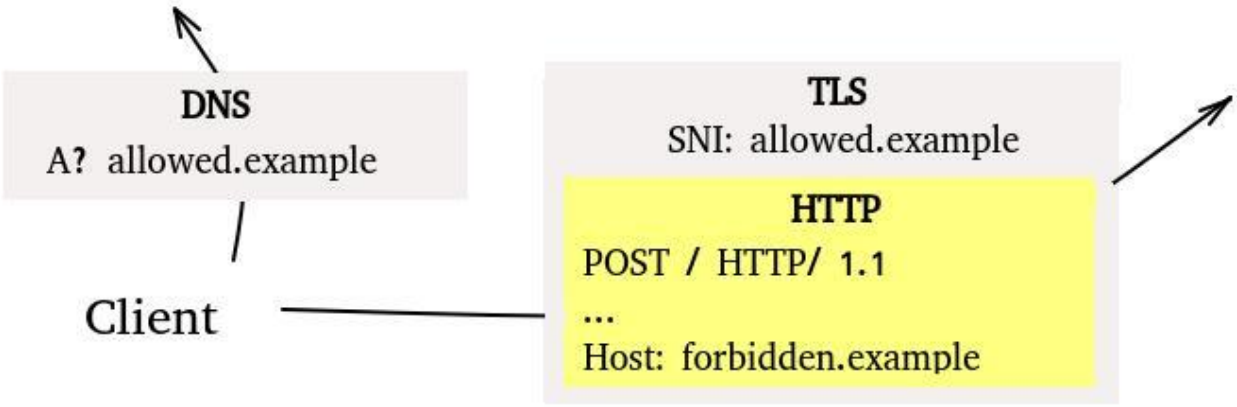

Рисунок 2.2 – Кроки клієнта для запиту інтернет-сторінки через фронтовий домен

Розширення SNI та заголовок «Host» слугують подібним цілям. Вони обидва дозволяють віртуальний хостинг, де один сервер обробляє запити для кількох доменів. Обидва поля дозволяють клієнту повідомляти серверу, до якого домену він хоче отримати доступ, але працюють вони на різних рівнях. SNI працює на рівні TLS, повідомляючи серверу, який сертифікат надсилати. Заголовок Host працює на рівні HTTP, повідомляючи серверу, який вміст обслуговувати.

Віртуальний хостинг у формі мереж доставки вмісту (CDN) зараз є загальним явищем. CDN працює, розміщуючи «крайовий сервер» між клієнтом і місцем призначення, що в цьому контексті називається «вихідним сервером». Коли граничний сервер отримує запит HTTP, він пересилає запит на вихідний сервер, на основі заголовку «Host». Крайовий сервер отримує відповідь від вихідного сервера і пересилає його назад клієнту. Крайовий сервер є фактично

проксі-сервером: клієнт ніколи не зв'язується з місцем призначення безпосередньо, а лише через посередницький CDN, який перешкоджає блокуванню адреси.

Вміст повідомлень клієнта, а також доменне ім'я справжнього пункту призначення захищені за допомогою шифрування TLS. Цензор може заблокувати крайові сервери CDN або фронтовий домен лише ціною блокування всього іншого, не пов'язаного з обходом трафіку на ці адреси, з побічною шкодою.

Спочатку може здатися, що фронтинг доменів корисний лише для доступу до веб-сайтів HTTPS. Але поширення ідеї роботи з довільними пунктами призначення вимагає лише незначного додаткового етапу запуску проксі-сервера на базі HTTPS та розміщення його на відповідній веб-службі. CDN переадресовує до проксі-сервера, який потім переадресовує до пункту призначення. Фронтинг домену захищає адресу проксі-сервера, який сам по собі не створює достатнього ризику побічної шкоди. Саме такий тип тунелювання HTTPS лежить в основі системи обходу, заснованої на фронтингу доменів, реалізація якої буде приведена далі.

Однією з найкращих особливостей фронтингу доменів є те, що він не вимагає жодної секретної інформації, повністю обходячи проблему розподілу проксі. Адреса крайового сервера CDN, адреса прихованого за ним проксі-сервера, той факт, що певна частина трафіку на крайовому сервері є обходом – все це може бути відоме цензору, не зменшуючи опір системі блокування. Це, звичайно, не означає, що фронтинг домену неможливо заблокувати – як завжди, здатність цензора блокувати залежить від його толерантності до побічної шкоди. Це лише змушує думати про обхід в цілому: не «чи не можна його заблокувати?», а «скільки коштує блокування?»

2.3 Система обходу Snowflake

Snowflake – це нова система обходу, яка базується на однорангових з’єднаннях через ефемерні (динамічні) проксі-сервери, що працюють у веб-браузерах. Проксі-сервери Snowflake легкі у використанні: проксі активується під час запиту веб-сторінки, а його вимкнення вимагає лише закриття вкладки браузера. Вони служать лише «тимчасовими східцями» для повноцінного проксі. Інтенсивність опору блокуванню залежить від наявності великої кількості проксі-серверів Snowflakes. Клієнт може використовувати певний проксі лише кілька

секунд або хвилин, перш ніж перейти на інший. Якщо цензору вдається заблокувати IP-адресу одного проксі, це не є перешкодою, оскільки багато інших тимчасових проксі готові зайняти його місце.

Snowflakes є наступником флеш-проксі, системи, яка аналогічно використовувала проксі-сервери на основі браузера, написаної на JavaScript. Флеш-проксі, з підтримкою obfs2 та obfs3, був одним із перших трьох, так званих, з'ємних траспортів (PT) для Tor, але з моменту його введення в 2013 році він ніколи не мав багато користувачів: використання базового протоколу флеш-проксі – WebSocket, вимагало від клієнтів дотримання складних інструкцій щодо переадресації портів [17]. З цієї причини флеш-проксі було припинено у 2016 році [17].

Snowflake зберігає основну ідею проксі-серверів у браузері, але замінює WebSocket на протокол комунікації в реальному часі (WebRTC), що представляє собою набір протоколів для однорангових комунікацій. Що важливо, WebRTC використовує протокол датаграм користувача (UDP) для передачі повідомлень, та включає засоби для обходу брандмауера з перетворенням мережевих адрес (NAT), що дозволяє більшості клієнтів використовувати його без ручного налаштування. WebRTC обов'язково шифрує дані, що як побічний ефект обфускує будь-які ключові слова або шаблони байтів у тунельованому трафіку.

Окрім флеш-проксі, найбільш схожою існуючою розробкою була колишня версія uProxy. uProxy вимагав від клієнтів «співучасника» за межами мережі цензора, який міг би керувати проксі-сервером; клієнт підключався через проксі-сервер використовуючи WebRTC; потім проксі-сервер безпосередньо отримував запитуваний клієнтом уніфікований вказівник ресурсу (URL). Snowflakes централізує процес виявлення проксі-сервера, усуваючи вимогу налаштовувати власний проксі-сервер поза брандмауером. Проксі-сервіси Snowflakes – це просто «німі канали» для більш працездатного проксі-сервера, що дозволяє їм направляти трафік, відмінний від веб-трафіку, і не дозволяє їм шпигувати за трафіком клієнта.

На рис. 2.3 зображена схема роботи Snowflakes.

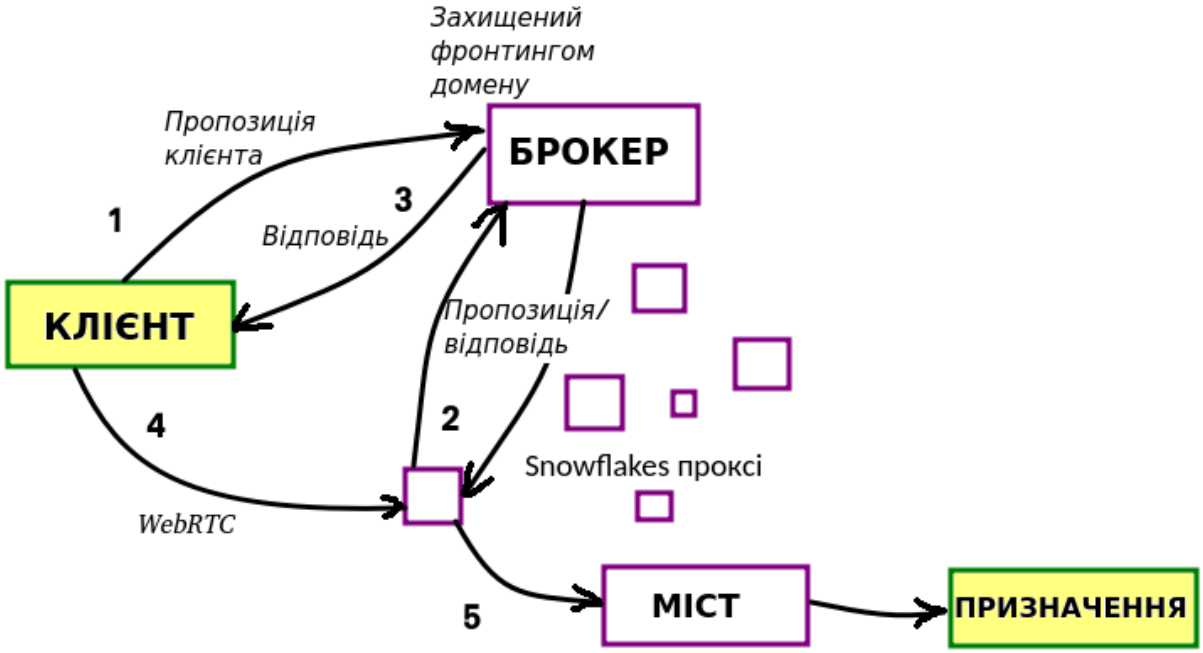

Рисунок 2.3 – Схематичне зображення роботи Snowflake

Є три основні компоненти системи Snowflake (рис. 2 3):

- багато проксі-серверів Snowflake, які «спілкуються» з клієнтами через WebRTC і перенаправляють їх трафік на міст;
- клієнти, відповідальні за запит на обслуговування, а потім встановлення однорангових зв’язків із проксі-серверами Snowflake;
- брокер, онлайн база даних, яка служить для встановлення відповідності клієнта із проксі-серверами Snowflake;
- міст, повнофункціональний проксі-сервер, здатний підключатися до будь-якого пункту призначення.

Робота архітектури залежить від вимог, щоб проксі працювали в браузері, і, щоб встановлення з’єднання забезпечувалися WebRTC, який використовує двостороннє рукостискання.

Підключення Snowflake відбувається в кілька кроків. На першому етапі, який називається «рандеву», клієнт та Snowflake обмінюються інформацією, необхідною для з'єднання WebRTC:

1) Клієнт реєструє свою потребу в послузі, надсилаючи повідомлення брокеру. Повідомлення, яке називається «offer» (з англ. «пропозиція»), містить IP-адресу клієнта та інші метадані, необхідні для встановлення з'єднання WebRTC.

2) У якийсь момент проксі-сервіс Snowflake з’являється в мережі та опитує

брокера. Брокер передає пропозицію клієнта проксі-серверу Snowflake, який надсилає свою відповідь, що містить його IP-адресу та інші метадані підключення, які клієнт повинен знати.

3) Брокер надсилає клієнту відповідь Snowflake. На цьому рандеву закінчено. Snowflake має пропозицію клієнта, а клієнт – відповідь Snowflake, тому вони мають всю інформацію, необхідну для встановлення зв’язку WebRTC між собою.

4) Клієнт та проксі-сервіс Snowflake з'єднуються між собою за допомогою WebRTC.

5) Проксі-сервер Snowflake підключається до мосту (за допомогою WebSocket; хоча конкретний тип каналу для цього кроку не має значення).

Потім проксі-сервер Snowflake передає дані між клієнтом та мостом, доки передача не буде припинена. Зв'язок клієнта з мостом шифрується та автентифікується через тунель WebRTC, тому проксі не може втручатися. Коли проксі-сервіс Snowflake «закінчується», клієнт може виконати запит нового. Етапи 1, 2 та 3 відбуваються синхронно, використовуючи чергування HTTP-запитів та відповідей (рис. 2.4).

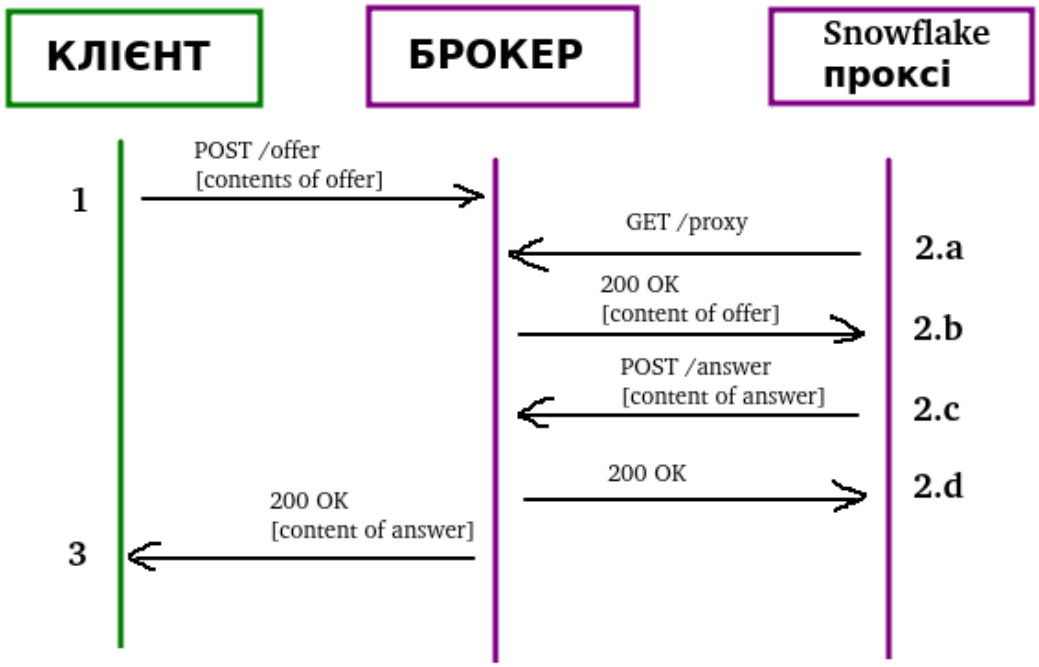

Рисунок 2.4 – Призначення рандеву Snowflake

У проміжку між запитом клієнта і відповіддю проксі-сервіс Snowflake робить дві власні пари запит-відповідь: перша призначена для вивчення пропозиції клієнта, а друга для надання відповіді.

При дослідженні схеми роботи Snowflake може виникнути запитання, якщо адресу брокера, яку, вважається, важко заблокувати (через розміщення його у провайдерів CDN), чи не достатньо цього для обходу? Відповідь полягає в тому, що цього, насправді, достатньо, але недоліком побудови системи виключно на фронтингу доменів є високі грошові витрати [18]. Snowflake «вивантажує» основну частину передачі даних на WebRTC і використовує дорогий фронтинг домену лише для призначення «рандеву».

Є дві причини, чому проксі Snowflake спрямовує клієнтський трафік на окремий міст, а не підключається безпосередньо до бажаного пункту призначення клієнта. Перша – це загальність: проксі-сервер на основі браузера може робити лише те, що може робити браузер; він може отримувати веб-сторінки, але не може, наприклад, відкривати сокети для довільних пунктів призначення. Друга – конфіденційність: проксі керують ненадійні, потенційно зловмисні незнайомці. Якби вони мали вихід до клієнтського трафіку безпосередньо, вони могли б його підробляти. Більше того, зловмисний клієнт міг би спричинити підключення добросовісного проксі-сервера до підозрілих пунктів призначення, потенційно спричиняючи проблеми у його власника.

2.4 Висновки до другого розділу

В даному розділі проаналізовано роботу активного зондування, як допоміжного механізму DPI-систем. Було досліджено принципи роботи таких методів боротьби з активним зондуванням як методу загального обходу, заснованого на HTTPS та систему обходу Snowflake. Таким чином було приведено необхідні умови для приховання факту використання обходу DPI-систем, що повинні бути дотримані при розробці власної програми, стійкої до активного зондування.

3 АНАЛІЗ ТА ДОСЛІДЖЕННЯ ВІДОМИХ ПРОГРАМ, СТІЙКИХ ДО DPI

3.1 Методика дослідження програм, стійких до DPI-аналізу

Для тестування програм, технічні рішення яких мають потенціал до обходу блокування ресурсів системою DPI необхідно, щоб трафік цих програм проходив через мережу з встановленою DPI-системою. Тут може існувати лише два рішення:

- налаштування власної DPI-системи;
- користування мережею, що вже має впроваджену систему DPI.

В якості допоміжної платформи у першому варіанті можна скористатися рядом програм, головною функцією яких є DPI-аналіз пакетів (наприклад, nDPI або OpManager). Комерційні продукти у вигляді апаратної реалізації не можуть бути розглянути через високу собівартість.

Такий спосіб є довготривалою та кропіткою справою, і не є метою цієї роботи. У другому способі мережами з системами DPI можуть виступати національні мережі з відносно суровими правилами користування Інтернетом. Так до країн, що нехтують принципом мережевого нейтралітету відносять Іран, Пакистан, Індію, Китай, Росію, Північну Корею, Туреччину, Тайланд та інші [19]. Придбавши віртуальний приватний сервер (VPS) в одній з таких країн з’являється можливість проведення експерименту з найбільш розвиненими DPI-системами. Під час пошуку серверу з’ясувалося, що в певних країнах (Китаї, Ірані, Єгипті та деяких інших) це зробити не можливо через політичні аспекти країн [20], а в вартість VPS в Туреччині, Киргизстані, Індії є високою в порівнянні з вартістю такого ж сервера в Україні. Серед доступних, але далеко не найкращих рішень, в Інтернеті можна знайти, так звані, proxy lists, що надають список безкоштовних проксі в безлічі країнах. Такі проксі зазвичай є нестабільними (через їхнє використання багатьма користувачами), і, що найважливіше, робота через них становить загрозу конфіденційності даних користувача [21].

Таким чином, в атестаційній роботі вирішено пропускати трафік через українського ISP, який за деякими джерелами має встановлену DPI-систему [22]. Функціонал такого DPI не такий широкий, як у китайських ISP [23], але в рамках власної роботи було вирішено скористатись саме цим рішенням. В залежності від типу трафіку, що визначається DPI-системою провайдер може надавати йому

певний пріоритет, що можна буде далі побачити у зміні пропускної здатності каналу. Так можна побачити, як працюють ті чи інші програмні реалізації обходу DPI в реальних умовах.

Протягом усього дослідження програми-клієнти були розташовані на власному комп'ютері з встановленою операційною системою (ОС) Linux. Програми-сервери були розташовані на віртуальному приватному сервері VPS, також, з ОС Linux. В рамках цієї роботи комп'ютер та VPS розташовувалися в Україні. На рис. 3.1 зображена структура мережі, що використовується в даному дослідженні.

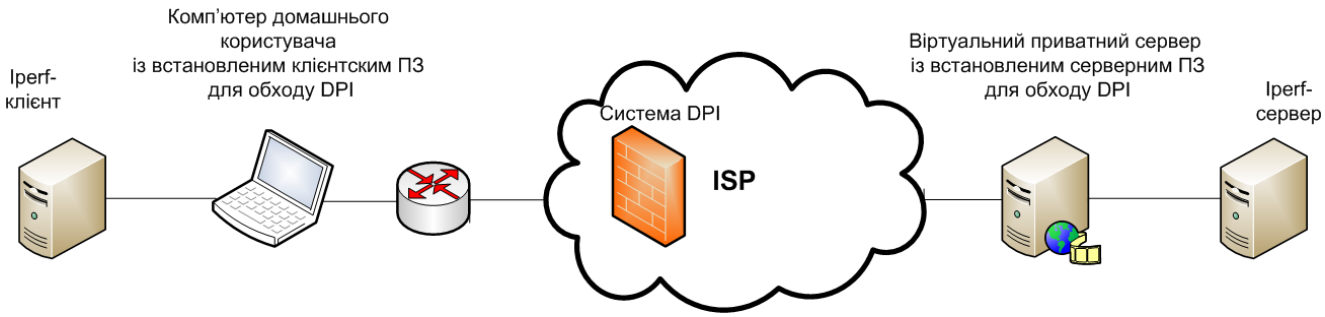

Рисунок 3.1 — Структура мережі

В роботі були оцінені значення параметрів встановленого з'єднання між програмою-клієнтом та програмою-сервером. Для цього використовувалася утиліта iperf3. Iperf – консольна утиліта з відкритим вихідним кодом, призначена для тестування пропускної здатності та інших параметрів мережі . З її допомогою досить просто виміряти максимальну пропускну здатність мережі між сервером і клієнтом або проводити тестування навантаження каналу зв'язку.

Дні та години, протягом яких проводяться виміри, мають безпосереднє значення на оцінку параметрів мережі. Опираючись на власний досвід моніторингу даної мережі було вирішено проводити досліди між 23:00 та 2:00. На основі статистичних даних, отриманих під час вимірювання, на рис. 3.2 зображено зміну пропускної здатності при прямому підключенні.

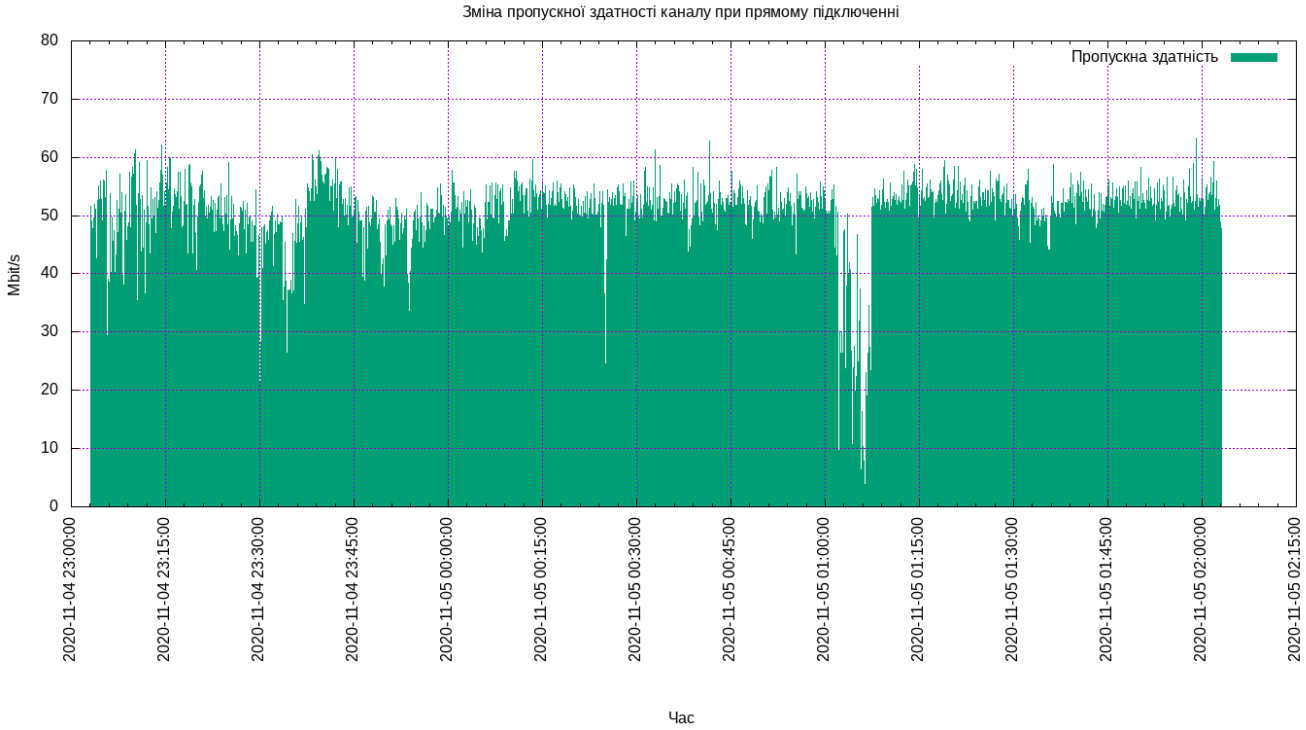

Рисунок 3.2 — Зміна пропускної здатності каналу при прямому підключенні

На рис. 3.2 видно, що трафік є відносно нестабільним. А на проміжку від 1:00 до 1:10 видно його тимчасове «придушення». Під час тестування середня швидкість становила 42,9 Мбіт. Кількість «перевідправлень» (втрачених сегментів TCP) становить 78 849 (через пошкодження пакетів або їх втрату при передачі через мережу Інтернет, що може бути результатом відповідної політики провайдера до такого типу трафіку)

В рамках даного розділу розглянуті та перевірені на базі приведеної мережі найпопулярніші рішення для обходу DPI-аналізу. Зокрема, тунелювання SSH (Secure Shell), VPN з підтримкою протоколу рівня захищених сокетів (SSL) та обфускація додатків.

3.2 SSH-тунелювання для приховання трафіку користувача

Основним завданням SSH є організація безпечного віддаленого доступу на основі мережі Інтернет. Цей стандартний протокол використовує аутентифікацію та шифрування для забезпечення конфіденційності та цілісності даних, якими обмінюються клієнт та сервер. Протокол SSH може стати у нагоді у випадку DPI-аналізу. Оскільки Інтернет провайдери не можуть блокувати SSH-трафік через

вагоме значення протоколу SSH для віддаленого адміністрування, то в якості методу стійкого до DPI можна вважати маскування трафіку користувача під SSH-трафік.

Протокол може тунелювати дані будь-якої програми, що використовує з’єднання TCP, із заздалегідь визначеним портом прослуховування. Тунелювання надає захист програмам, які в іншому випадку передаватимуть незахищений трафік через загальнодоступні мережі. Повідомлення додатків, передані з одного кінця з'єднання Secure Shell на інший, захищені криптографічними методами, що були узгоджені для цього з'єднання. Оскільки декілька програм можна мультиплексувати через одне з'єднання Secure Shell, фільтри брандмауера та маршрутизатора можуть бути зведені лише до одного вхідного порту: порту SSH(22).

Потоки програм тунелюються через SSH шляхом перенаправлення окремих портів TCP. У даній роботі тунелювання буде виконане за допомогою локальної переадресації портів : тунелювання, що ініційоване клієнтом SSH. Цей напрямок є більш поширеним, ніж віддалена переадресація портів: тунелювання, що ініційоване сервером SSH.

Коли перенаправляється локальний порт (localport), клієнт SSH прослуховує вказаний порт TCP на локальному хості (localhost), а сервер SSH відкриває TCP-з'єднання з віддаленим хостом (remotehost). Прийнято, що:

- localhost відноситься до хосту клієнтського додатка; remotehost відноситься до хосту сервера додатків. Як правило, якщо localhost не вказано, він за замовчуванням визначає хост клієнта. Якщо Remotehost не вказано, він за замовчуванням визначає хост сервера;
- localport відноситься до порту , через який клієнтський додаток посилає повідомлення (порт, що прослуховує SSH-клієнт). remoteport відноситься до порту , який прослуховує SSH-сервер і, відповідно, сервер додатків. У більшості випадків локальним портом може бути будь-який довільний невикористаний порт на локальному хості. remoteport може бути призначений адміністрацією адресного простору Інтернет (IANA) і слухати порт додатка через тунель.

Щоб використовувати порт переадресації, в атестаційній роботі переналаштовано клієнтську програму на підключення до «localhost:localport» замість «remotehost:remoteport». Пакети, надіслані клієнтом на «localhost:localport» , перехоплюються SSH-клієнтом, шифруються та тунелюються через з'єднання із

SSH-сервером. Після отримання сервером даних, він розшифровує ці пакети, передаючи їх у вигляді «чистого тексту» через TCP-з'єднання із службою на «remotehost:remoteport» . Локальна переадресація за допомогою SSH зображена на рис. 3.3.

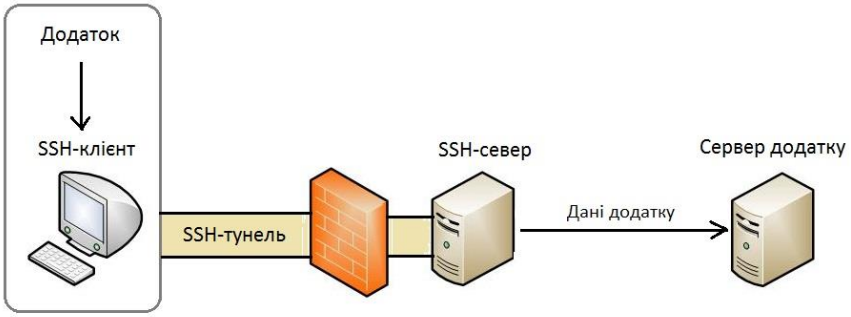

Рисунок 3.3 — Локальна переадресація портів за допомогою SSH

Транзитний трафік між SSH-клієнтом та SSH-сервером захищені криптографічно. Однак трафік між SSH-сервером та віддаленою мережею ні. Як правило, SSH-сервер знаходиться всередині периметра мережі, за брандмауером. Брандмауер налаштований для дозволу протоколу SSH, але не для тунельованих протоколів додатків. По суті, ця конфігурація використовує брандмауер для захисту трафіку та внутрішніх серверів у надійній локальній мережі.

SSH-сервер також може працювати на тій самій машині, що і серверна програма (рис. 3.2). У цьому випадку немає необхідності вказувати віддалений хост: SSH-клієнт та SSH-сервер взаємодіють із програмами на кожному локальному хості.

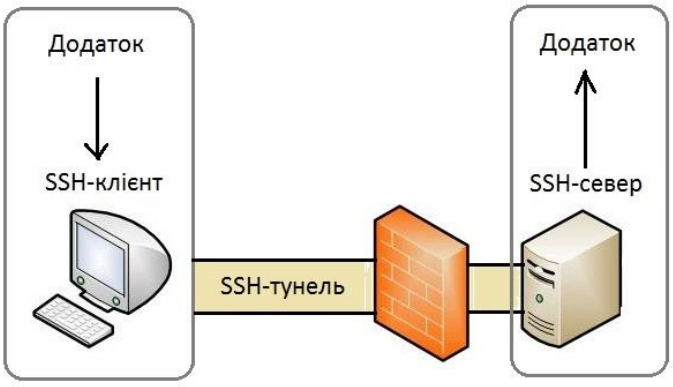

Рисунок 3.4 — Локальна переадресація портів до програм

Даний метод передачі контексту має певні аспекти, які слід зазначити:

- в операційній системі (ОС) Linux SSH вбудований в бібліотеку OpenSSH, і доступний у більшості його дистрибутивах. Також протокол доступний в MacOS. Однак, наприклад, в ОС Windows для користування протоколом потребується встановлення додаткового програмного забезпечення;
- хоча це концептуально можливо, стандартний SSH не пересилає датаграми UDP.

Приклад команда для встановлення тунелю від клієнта до SSH-сервера виглядає наступним чином:

#ssh -L localport:localport:remoteport <IP-адреса ssh-серверу>.

Після цього буде створене з’єднання, через яке будуть передаватися зашифровані дані. Для перенаправлення даних додатка (в нашому випадку тестові пакети iperf3) на клієнтській стороні виконується команда:

#iperf3 -c localhost -p localport .

Протягом 3 годин була запущена утиліта iperf3 для виміру параметрів мережі між клієнтом та сервером. Зміну пропускної здатності при підключенні по SSH-тунелю через 22 порт приведено на рис. 3.5. За час вимірювання 5258 пакетів було перевідправлено сервером-iperf3. Середнім значення пропускної здатності є 45,6 Мбіт/с.

Рисунок 3.5 — Зміна пропускної здатності при підключенні по SSH-тунелю через 22-ий порт

Розглядаючи рис. 3.5 та порівнюючи його з трафіком на рис. 3.2 можна сказати, що передача SSH-трафіку стабільніша. Існування SSH-тунелю відобразилося на середній швидкості передачі даних: так вона становить майже на 3 Мбіт/с більше, ніж при підключенні на пряму.

Крім цього в роботі була виміряна пропускна здатність при підключенні по SSH-тунелю через нестандартний порт (63366). Результат вимірювання приведено на рис. 3.6.

Рисунок 3.6 — Зміна пропускної здатності при підключенні по SSH-тунелю через нестандартний порт

На проміжку від 23:00 до 23:50 (рис. 3.6) можна спостерігати стабільний трафік з невеликим приростом швидкості (швидкість близько 55 Мбіт/с). Після цього йде значний занепад до 0 Мбіт/с, що, ймовірно, відбувається при переключенні маршруту ISP. Далі, після близько години часу, трафік місцями значно «просідає» у швидкості. А майже в кінці вимірювання можна спостерігати щось схоже на лавоподібний трафік. Середня швидкість становить 40,5 Мбіт/с.

3.3 Virtual Private Network над Secure Sockets Layer

VPN — технологія, що добре підходить у випадку, коли потрібно приховати справжню адресу призначення. Проте більшість VPN-рішень може бути розпізнано через особливість їх трафіку [24]. Так системи DPI, що володіють технологією DPC можуть аналізувати трафік протягом певного часу (годин, днів або тижнів [25]), достатнього для прийняття рішення щодо використання VPN-додатків. І на основі цього приймати певну політику (розірвання з'єднання, сповільнення трафіку).

Однак існують VPN-додатки, що передають трафік на базі протоколу SSL/TLS. Це робить трафік користувача схожим на трафік HTTPS (аналіз якого приведено на рис 3.7). І тому DPI розпізнати його стає набагато важче [26]. В атестаційній роботі в якості прикладу розглянуто та протестовано OpenConnect VPN.

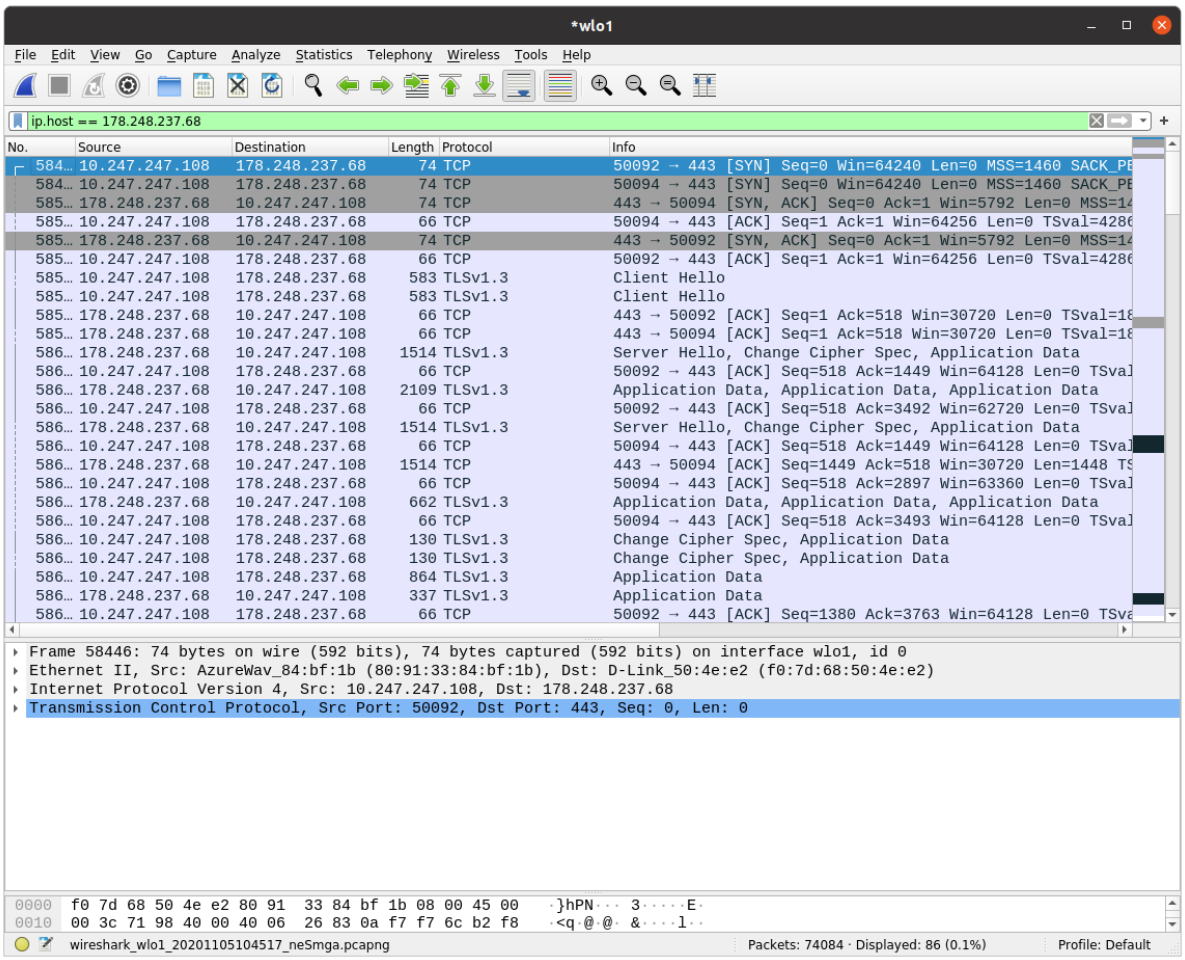

Рисунок 3.7 — Аналіз трафіку HTTPS в програмі Wireshark

OpenConnect – програма з відкритим кодом що реалізує технологію віртуальної приватної мережі для встановлення безпечних з'єднань типу «точка-точка». Програма 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. З точки зору обходу систем DPI перевагою цього VPN є його здатність передавати трафік користувача за допомогою протоколу TLS/SSL (рис. 3.8)

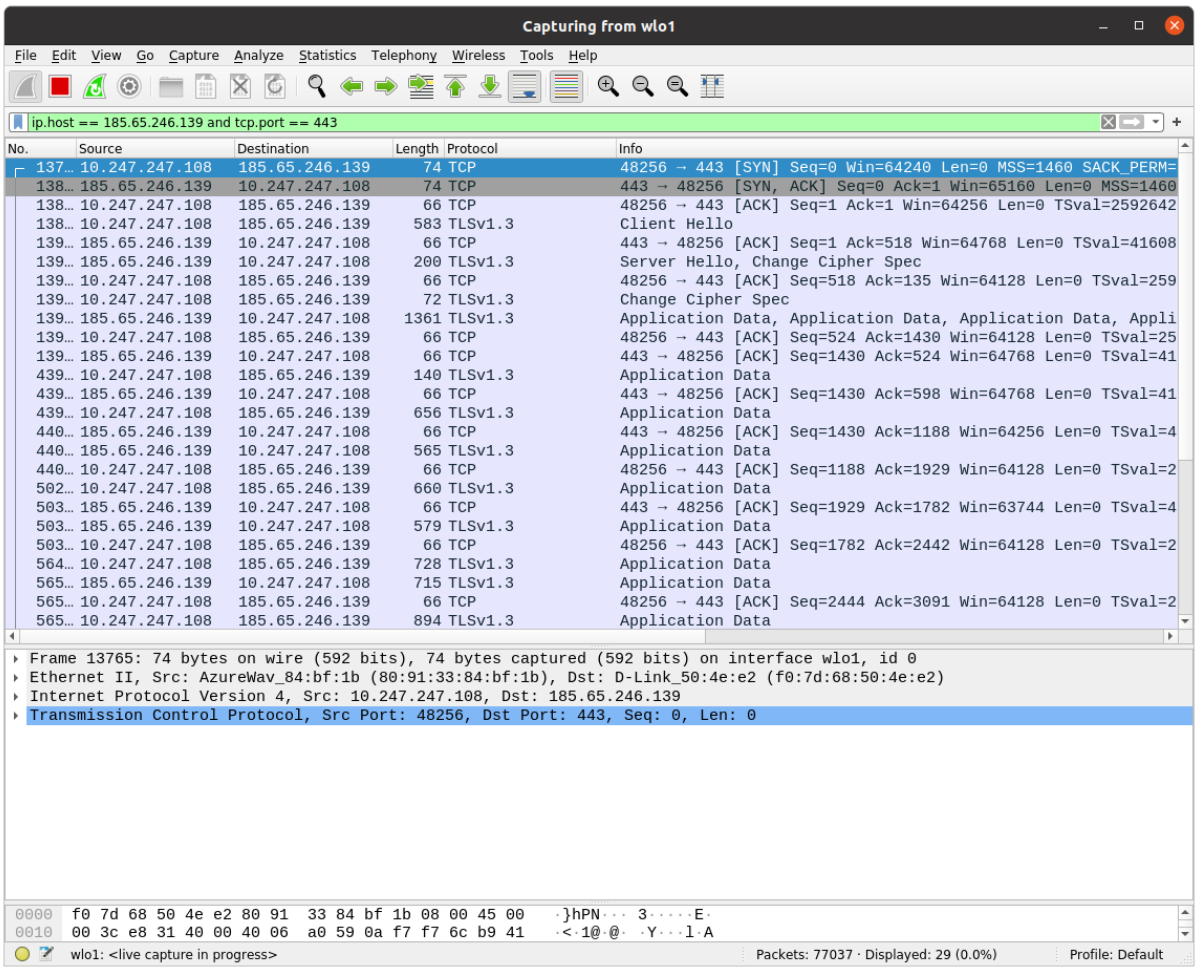

Рисунок 3.8 — Аналіз трафіку OpenConnect в програмі Wireshark

Встановлення з'єднання відбувається у дві фази.

Спочатку ініціюється з'єднання HTTPS, через яке користувач автентифікується (за допомогою сертифікату, паролю, SecurID, тощо). Після автентифікації користувач отримує від серверу OpenConnect файл «cookie», який можна використовувати для з'єднання VPN (тунелю).

Другий етап використовує файл «cookie» для підключення до тунелю через HTTPS для передачі пакетів даних через встановлене з'єднання. По можливості також налаштовується UDP-тунель: AnyConnect/Openconnect використовує у якості транспорту TLS датаграми (DTLS). Тунель UDP є гарним рішенням з міркувань продуктивності мережі (рішення типу «TCP над TCP» працює погано і є ненадійним [27]). Проте використання DTLS може слугувати індикатором використання користувачем технології VPN.

В рамках даного дослідження мною було вирішено протестувати протокол OpenConnect у двох режимах: TLS та DTLS.

Після проведення налаштування клієнта OpenConnect (openconnect) та сервера OpenConnect (ocserv) було встановлене з’єднання.

Результати вимірювання пропускної здатності між клієнтом та сервером лише в режимі TLS зображено на рис. 3.9.

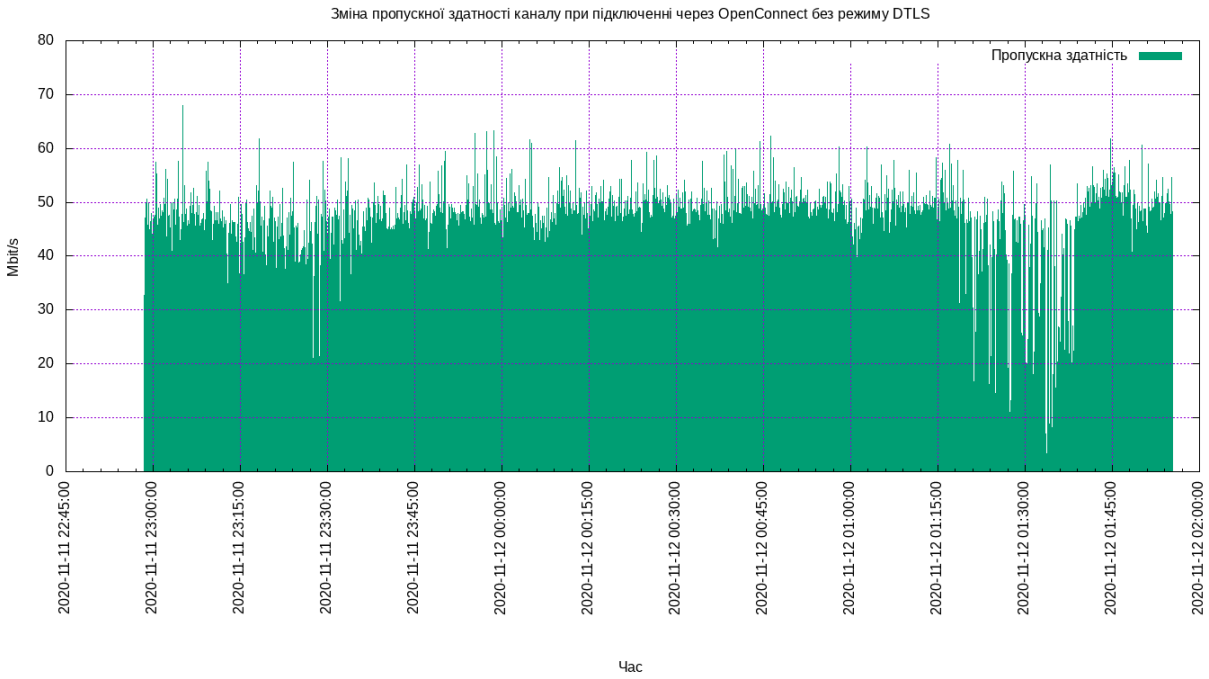

Рисунок 3.9 – Зміна пропускної здатності каналу при підключенні через OpenConnect в режимі TLS

Результати вимірювання пропускної здатності між клієнтом OpenConnect та сервером ocserv в режимі DTLS зображено на рис. 3.10.

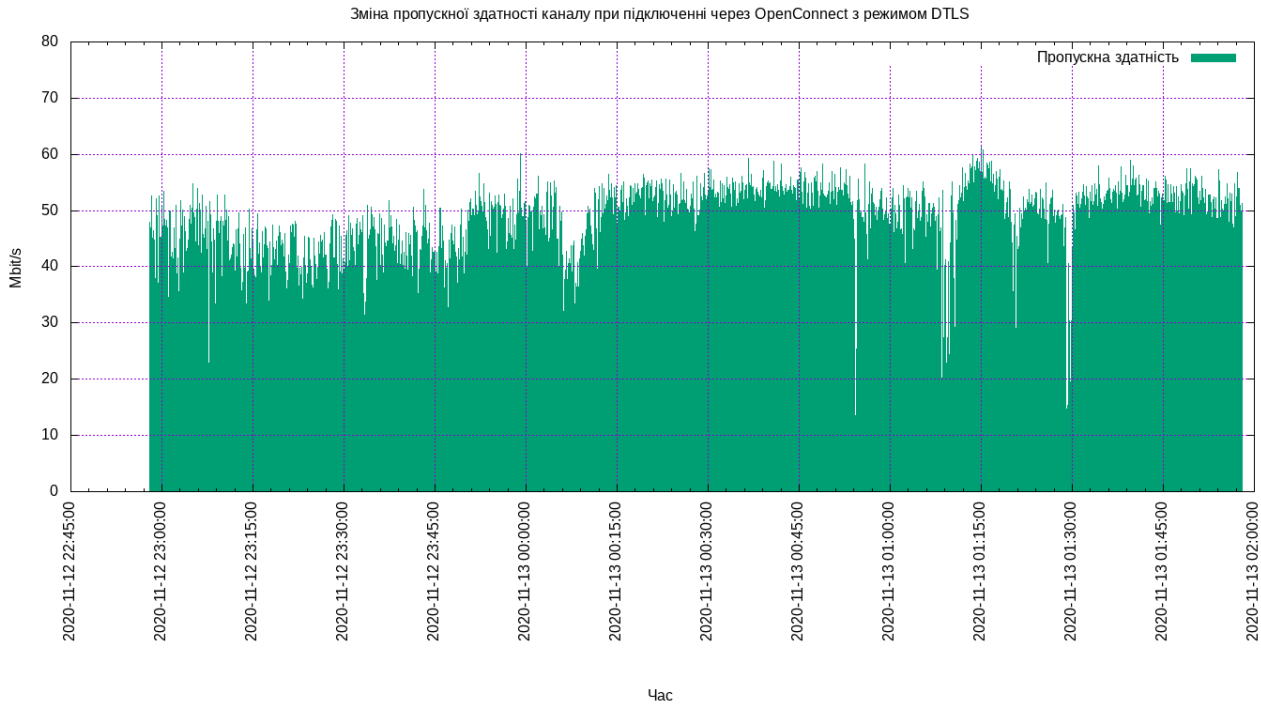

Рисунок 3.10 – Зміна пропускної здатності каналу між клієнтом OpenConnect та сервером ocserv в режимі DTLS

У першому випадку (рис. 3.9) середнє значення швидкості передачі даних становить 39, 3 Мбіт/с, а у другому (рис. 3.10) – 40, 3 Мбіт/с. Кількість втрачених сегментів TCP в обох вимірюваннях приблизно однакова й становить близько 85 000.

Трафік на обох рисунках можна назвати відносно стабільним; лише з короткочасними «просіданнями» швидкості в деяких відрізках часу. На рис. 3.9 можна спостерігати навіть деякі «підйоми» швидкості від 60 Мбіт/с на усьому проміжку вимірювання (що, скоріше за все, пов’язано з особливістю навантаження каналів зв’язку провайдера). На інтервалі часу від 1:20 – 1:40 помітні різкі «стрибки» трафіку вгору-вниз, що може пояснюватися лише додатковим навантаженням каналу між клієнтом та сервером.

3.3 Обфускація трафіку додатків користувача

Читаючи про функції різних VPN, можна зустріти терміни, пов'язані з їх маскуванням. Є багато «модних слів», пов'язаних з цією темою, але багато які з них означають приблизно одне й те саме. Деякими прикладами є «прихований

VPN» або «прихований режим», «технологія маскування» і подібне. Все це означає, що VPN використовує певний спосіб заплутування, щоб замаскувати трафік, коли використовуються відповідні налаштування.

Далі будуть приведені два методи, які можуть використовуватися для обфускації трафіку. Слід звернути увагу, що для роботи будь-якого виду обфускації необхідно налаштувати клієнт і сервер для його використання.

Для тестування обох способів обфускації в атестаційній роботі вирішено використати в якості додатку протокол OpenVPN, який, як виявилося, може легко детектуватися системами DPI [28]. Слід також зазначити, що OpenVPN в усіх дослідах буде працювати поверх TCP-протоколу. На рис. 3.11 зображено зміну пропускної здатності каналу при підключенні до сервера OpenVPN без маскування (використовується зарегестрований порт 1194, що належить OpenVPN).

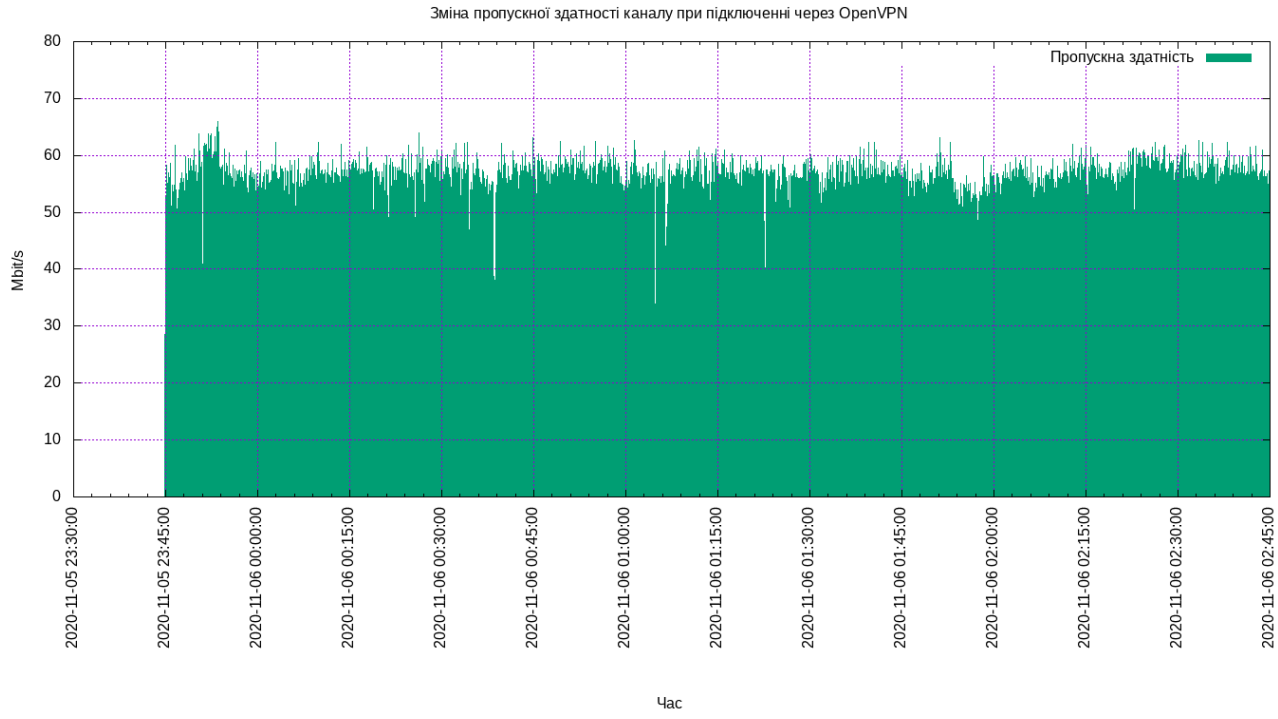

Рисунок 3.11 – Зміна пропускної здатності каналу мережі OpenVPN

Трафік є досить стабільним, однак в деяких інтервалах часу спостерігаються короткочасні «просідання» швидкості до 40 та нижче Мбіт/с (23:50, 0:40, 1:05, 1:27) . Середня швидкість передачі даних в мережі із звичайним підключення OpenVPN підвищилася до 51,3 Мбіт/с.

Кількість «перевідправлень» становить 77253.

3.3.1 Програма для обфускації трафіку на базі Pluggable Transports

З'ємні транспорти (з англ. Pluggable Transports, PT) є підпроєктом проєкту Tor (анонімний браузер Tor). Він був створений у відповідь на блокування трафіку Tor в деяких країнах [29]. Він заплутує трафік, так щоб він не був розпізнаним і не підпадав під жодну категорію DPI-аналізу. І хоча PT був розроблений для використання браузера Tor, його також можна використовувати з іншими додатками.

Щоб зрозуміти, які саме функції виконує PT, в атестаційній роботі досліджено дані трафіку на прикладі Tor.

На рис. 3.12 зображений дамп повідомлення «Client Hello», що відправляється клієнтом при ініціалізації з'єднання з вхідним вузлом мережі Tor (без PT). Виділеними позначені дані, що можуть бути цікавими для нас (зокрема, для цензора): список шифрів, назва серверу, розширення TLS.

```
00000000  16 03 01 00 e5 01 00 00  e1 03 03 d2 23 9e 55 3d  ........ ....#.U=
00000010  21 cc 80 d5 2b 34 4b e1  be 90 04 c6 d0 f8 f9 a5  !...+4K. ........
00000020  56 3f 01 22 ef c6 e1 22  64 c5 0a 00 00 30 c0 2b  V?."...." d....0.+
00000030  c0 2f c0 0a c0 09 c0 13  c0 14 c0 12 c0 07 c0 11  ./...... ........
00000040  00 33 00 32 00 45 00 39  00 38 00 88 00 16 00 2f  .3.2.E.9 .8...../
00000050  00 41 00 35 00 84 00 0a  00 05 00 04 00 ff 01 00  .A.5.... ........
00000060  00 88 00 00 00 17 00 15  00 00 12 77 77 77 2e 6b  ........ ...www.k
00000070  66 33 69 61 6d 6d 6e 79  70 2e 63 6f 6d 00 0b 00  f3iammny p.com...
00000080  04 03 00 01 02 00 0a 00  34 00 32 00 0e 00 0d 00  ........ 4.2.....
00000090  19 00 0b 00 0c 00 18 00  09 00 0a 00 16 00 17 00  ........ ........
000000A0  08 00 06 00 07 00 14 00  15 00 04 00 05 00 12 00  ........ ........
000000B0  13 00 01 00 02 00 03 00  0f 00 10 00 11 00 23 00  ........ ......#.
000000C0  00 00 0d 00 20 00 1e 06  01 06 02 06 03 05 01 05  .... ... ........
000000D0  02 05 03 04 01 04 02 04  03 03 01 03 02 03 03 02  ........ ........
000000E0  01 02 02 02 03 00 0f 00  01 01                    ........ ..
```

Рисунок 3.12 – Аналіз повідомлення Client Hello без використання PT

Читати такі повідомлення легше після їхнього розбору аналізатором Wireshark. На рис. 3.13 виділені значення: список шифрів, назва серверу, розширення TLS після розбору повідомлення аналізатором.

```
Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 225
    Version: TLS 1.2 (0x0303)
    Random
        gmt_unix_time: Sep 19, 2081 16:20:53.000000000 PDT
        random_bytes: 3d21cc80d52b344be1be9004c6d0f8f9a5563f0122efc6e1...
    Session ID Length: 0
    Cipher Suites Length: 48
    Cipher Suites (24 suites)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Cipher Suite: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)
        Cipher Suite: TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
        Cipher Suite: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
        Cipher Suite: TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x0045)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
        Cipher Suite: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
        Cipher Suite: TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x0088)
        Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x0041)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Cipher Suite: TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x0084)
        Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
        Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
        Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
        Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
    Compression Methods Length: 1
    Compression Methods (1 method)
        Compression Method: null (0)
    Extensions Length: 136
    Extension: server_name
        Type: server_name (0x0000)
        Length: 23
        Server Name Indication extension
            Server Name list length: 21
            Server Name Type: host_name (0)
            Server Name length: 18
            Server Name: www.kf3iammnyp.com
    Extension: ec_point_formats
```

Рисунок 3.13 – Аналіз розбору Client Hello без використання PT

Цікавою частиною повідомлення для цензорів у свій час виявився перелік наборів шифрів. Великий китайський брандмауер заблокував Tor у 2011 році на основі особливого списку шифру Tor [30]. У відповідь на що, Tor змінив свій список набору шифрів на такий самий, як і список у Firefox.

Сформоване ім’я сервера було згенеровано клієнтом і є випадковим значенням – www.kf3iammnyp.com.Сервер, у свою чергу, у якості відповіді

відправляє «Server Hello», де зазначає вибраний метод шифрування та згенероване власне ім'я, яке відображається у його сертифікаті. Усе з першого погляду виглядає, як звичайний HTTPS запит, а не спроба підключення до вузла Tor. Однак у цьому рукостисканні дані заголовків піддаються аналізу DPI.

Наприклад, хоч імена не є фіксованими байтовими шаблоном, можна проаналізувати заголовки, переглянувши кілька рукостискань і переконатись, що імена завжди відповідають очікуваному шаблону, як це робить сценарій Bro [31].

Obfs2 був першим PG. За допомогою нього трафік виглядає як випадковий потік байтів для спостерігача. Він був втілений розробниками, натхненими obfuscated-openssh, а наступниками програми стали obfs3, ScrambleSuit та obfs

На сьогодні obfs2 застарілий, тому що його може ідентифікувати абсолютно пасивний спостерігач (як ми побачимо, жоден MitM для цього не потрібен). Ідея obfs2 в основному полягає в наступному:

- надіслати ключ шифрування, потім «магічне» число, довжину заповнення (padlen) та деяку кількість заповнення (padding), зашифровані цим ключем;
- визначити новий ключ як з клієнтського, так і з початкового ключа сервера та використовувати його для шифрування всіх даних програми.

І клієнт, і сервер виконують однакові операції, відрізняючись лише кількома константами. На рис 3.14 та 3.15 зображено байти повідомлення початку сеанса obfs2 з клієнтської та серверної сторони. Виділені байти представляють відповідно ключ, «магічне число», padlen, padding.

```
00000000  1c f1 11 91 af fb 8f 31  08 c7 a6 53 c3 84 10 a1  .......1 ...S....
00000010  e5 d1 ec bb 72 68 40 27  25 62 bb 8b c2 5b 57 1a  ....rh@' %b...[W.
00000020  f5 04 17 c3 8c 4d 99 9a  b9 3f a7 41 d8 99 61 88  .....M.. .?.A..a.
00000030  f8 71 16 66 d3 a3 20 1a  71 f1 ec 44 95 e7 07 05  .q.f.. . q..D....
00000040  0b cd 90 75 72 3f f9 37                           ...ur?.7
```

Рисунок 3.14 – Байти повідомлення початку сенсу для обфускації трафіку за допомогою PT obfs2 на клієнтській стороні від серверу

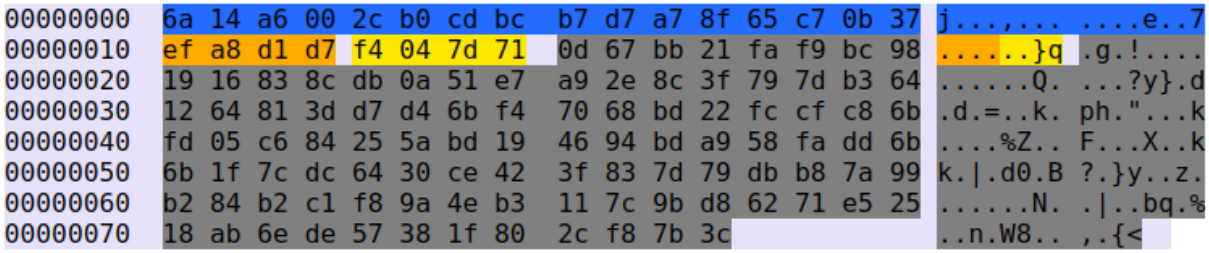

Рисунок 3.15 – Байти повідомлення початку сенсу для обфускації трафіку за допомогою PT obfs2 на серверній стороні від клієнта

В атестаційній роботі розгялнуто лише повідомлення, що було відправлено від obfs2-серверу obfs2-клієнту (рис.3.14)

З перших 16 байт (в даному випадку 1cf11191affb8f3108c7a653c38410a1) вилучається ключ та початкове значення режиму для режиму AES_CTR [32].

В додатку А приведена програма на мові Python для самостійної розшифровки перших зашифрованих 16 байт (e5d1ecbb726840272562bb8bc25b571a), результатом якої вийде значення 2bf5ca7e00000030eef53458d509c177. Перші 4 байта, 2bf5ca7e, завжди однакові. Це, так зване, «магічне» число визначає потік obfs4 (там написано «2bfscate» (to obfusctate)). Наступні 4 байта, 00000030, визначають кількість байт заповнення. В нашому випадку, 0x30 = 48 байт заповнення (зазвичай відступ – це випадкова довжина від 0 до 8192 байт). Наступні 48 байтів, починаючи з eef53458d509c177, є просто випадковим заповненням. Сервер виконує ідентичні операції.

Після цього обидві сторони переходять на нові ключі та значення лічильника, що були визначені з початкових значень. Ці атрибути будуть необхідні для розшифровки трафіка, що буде передаватися.

Таким чином, заголовки додатків (сигнатури) будуть зашифровані під час передачі пакетів через мережу Інтернет, і DPI не зможе їх використати, щоб визначити який додаток може передавати дані.

PT obfs3 усуває вразливість obfs2 до пасивної ідентифікації, додаючи обмін ключами Діффі-Хеллмана. Тепер для активної ідентифікації obfs3 потрібна активна атака людини посередині (MitM) або активне зондування. Існує каверза у обміні ключами Діффі-Хеллмана: він повинен виглядати рівномірно випадковим, як і решта протоколу. У звичайному випадку можна б було працювали у підгрупі Z_p для якогось безпечного простого числа p. Але лише приблизно половина цілих чисел від 0 до p знаходиться в Z_p.[33] Цензор міг би спостерігати за

повідомленнями обміну ключами і бачити, що цифри, які надсилаються, насправді не випадкові, а завжди знаходяться у вибраній підгрупі. Через деякий час цензор вирішить, що ви обмінюєтесь ключами, а не надсилаєте випадкові байти. obfs3 вирішує цю проблему за допомогою алгоритму, який називається UniformDH. Це невелике нововведення до звичайного Діффі-Хеллмана. Алгоритм роботи цього протоколу описано в наступному джерелі [34]. За допомогою UniformDH цензор не може визначити різницю між відкритим ключем та випадковим набором байтів.

В obfs3, як і в obfs2, і клієнт, і сервер виконують по суті однакові операції. Обидві сторони починають з надсилання свого відкритого ключа UniformDH з подальшим випадковим заповненням. На рис 3.16 зображено дамп повідомлення, відправленого від сервера клієнту, що використовують в якості PT obfs3.

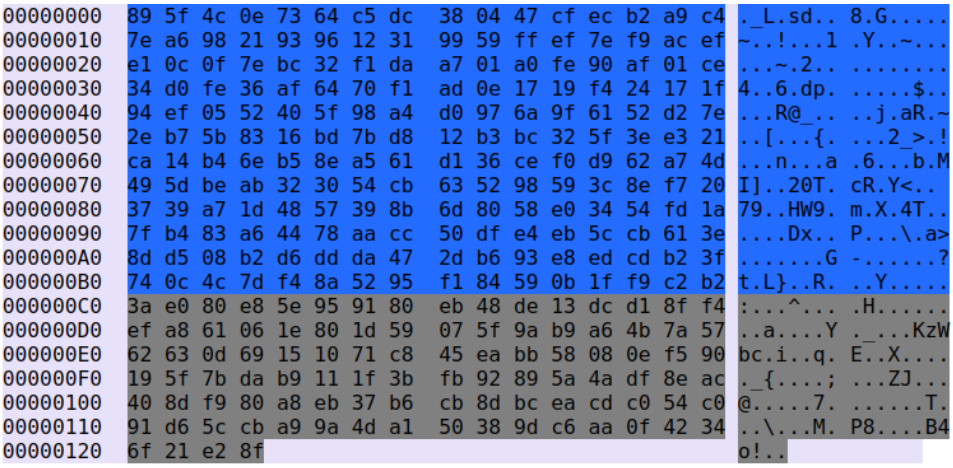

Рисунок 3.16 – Байти повідомлення obfs3

Перші 192 байти з кожної сторони – це відкриті ключі. Після цього виконується випадкова кількість заповнення (від 0 до 4097 байт). В нашому випадку сервер надіслав 100 байт заповнення. На відміну від obfs2, padlen не надсилається як ціле число: кінець заповнення позначений хеш-кодом автентицікації повідомлень (HMAC) на секреті, яким обидві сторони діляться після обміну ключами.

На рис. 3.17 зображений дамп повідомлення клієнта після обміну ключами.

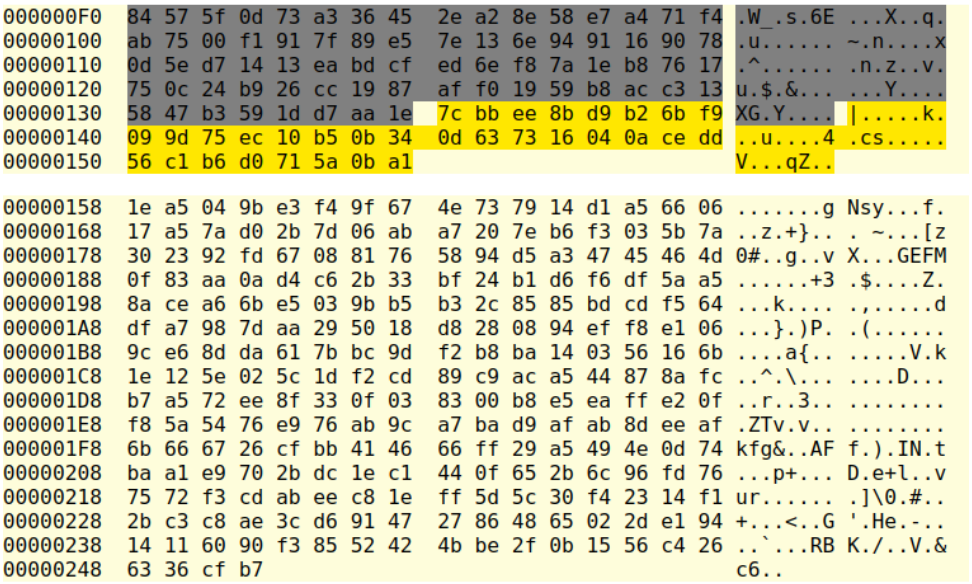

Рисунок 3.17 – Байти повідомлення з корисним навантаженням при використанні obfs3

Тепер, коли клієнт надсилає дані, він відправляє ще один блок випадкового заповнення між 0 і 4097 байтами, за яким слідує позначка кінця заповнення (7cbbee8bd9b26bf9099d75ec10b50b340d637316040acedd56c1b6d0715a0ba1), після чого слідують зашифровані дані корисного навантаження, зашифровані за допомогою ключа, отриманого із загального секрету.

У своїй відповіді сервер також надсилає ще кілька байтів заповнення та власний маркер закінчення заповнення, а потім зашифровані дані корисного навантаження.

Ні obfs2, ні obfs3 нічого не роблять, щоб замаскувати розміри та розрахунок часу відправлення пакетів. На рис. 3.18 можна бачити, що, крім різних рукостискань вгорі, розмір пакетів в основному незмінний (у прикладах використовується коротке заповнення, але на практиці заповнення obfs2 та obfs3 може становити до 8 кілобайт.).

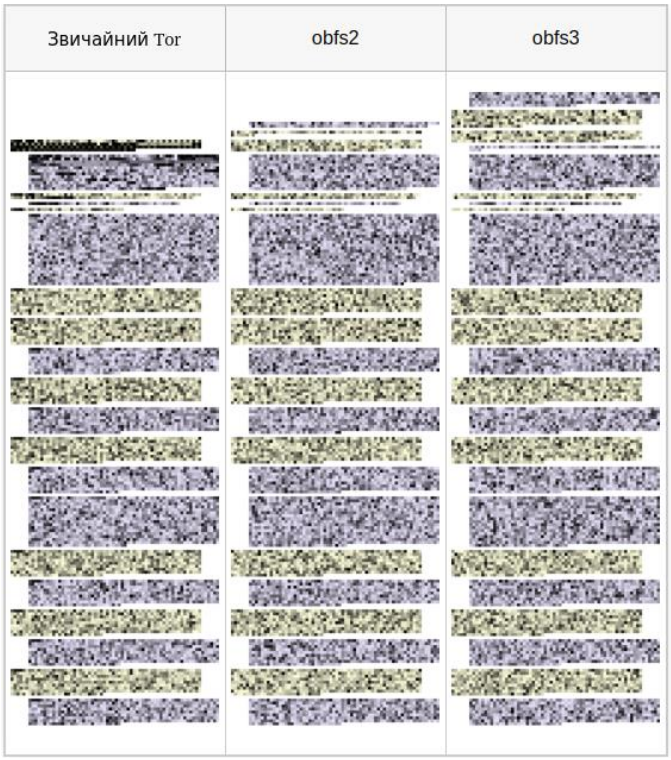

Рисунок 3.18 – Порівняння трафіку Tor без та з використанням obfs2 та obfs3

Якщо уважно придивитися до звичайного прикладу Tor, можна побачити темний мазок на початку кожного пакета – це заголовок простого тексту TLS, який PT має на меті заплутати для DPI.

ScrambleSuit – ще один PT, який виглядає як набір випадкових байтів. У нього є кілька нововведень над obfs3: він рандомізує розмір і час пакетів, а сервер чинить опір активному зондуванню, вимагаючи від клієнта секретного ключа, перш ніж він відповість. Дамп повідомлення виглядає як випадковий набір байтів, як і слід було очікувати (рис. 3.19).

```
00000000  00 90 a0 58 e3 05 c2 d6  f0 6e 86 5c 5d dd 57 1c ...X.... .n.\].W.
00000010  cf a0 ec ed 15 ed 00 e5  ed d6 d9 29 d9 31 ba 64 ........ ...).1.d
00000020  15 8e bd 84 93 9a e4 f8  c9 d4 ed 25 73 74 e5 69 ........ ...%st.i
00000030  2e 20 02 9a 8d 7c aa 56  04 d7 a7 74 47 31 f6 a3 . ...|.V ...tG1..
00000040  5b ae b2 36 78 57 05 2f  c5 9d 2b 20 7f 5f 8b 23 [..6xW./ ..+ ._.#
00000050  82 5a 47 6a 59 8e 2f 12  14 6a 85 ce 89 16 2a 42 .ZGjY./. .j....*B
00000060  b1 ea d8 3e 2d 92 e1 f3  f9 4d 8b 02 32 a2 c7 e0 ...>-... .M..2...
00000070  85 83 8a 29 6f 47 60 de  25 fb 0c c0 a8 37 55 38 ...)oG`. %....7U8
00000080  69 f5 95 24 5b 66 2c c5  3d 70 66 a5 a4 1b ea dc i..$[f,. =pf.....
00000090  44 f9 28 c2 d4 77 55 ff  49 2b 7d d5 ac 77 09 5b D.(..wU. I+}..w.[
000000A0  07 0c cb 80 8c 6b ca 20  e6 d5 f2 58 f3 69 c9 ac .....k.  ...X.i..
000000B0  da 3f 4c 46 ba 9c e7 c8  34 8d 8e 45 eb c0 35 d7 .?LF.... 4..E..5.
000000C0  98 e7 5f 92 d1 dc 9b f4  92 d4 04 cc 42 04 2c 0d ..._.... ....B.,.
000000D0  e0 9a                                          ..
```

Рисунок 3.19 – Байти повідомлення додатку, який неможливо визначити через його обфускацію

На рис. 3.20 зображена візуалізація завантаження веб-сторінки за допомогою звичайних Tor, obfs3 та ScrambleSuit.

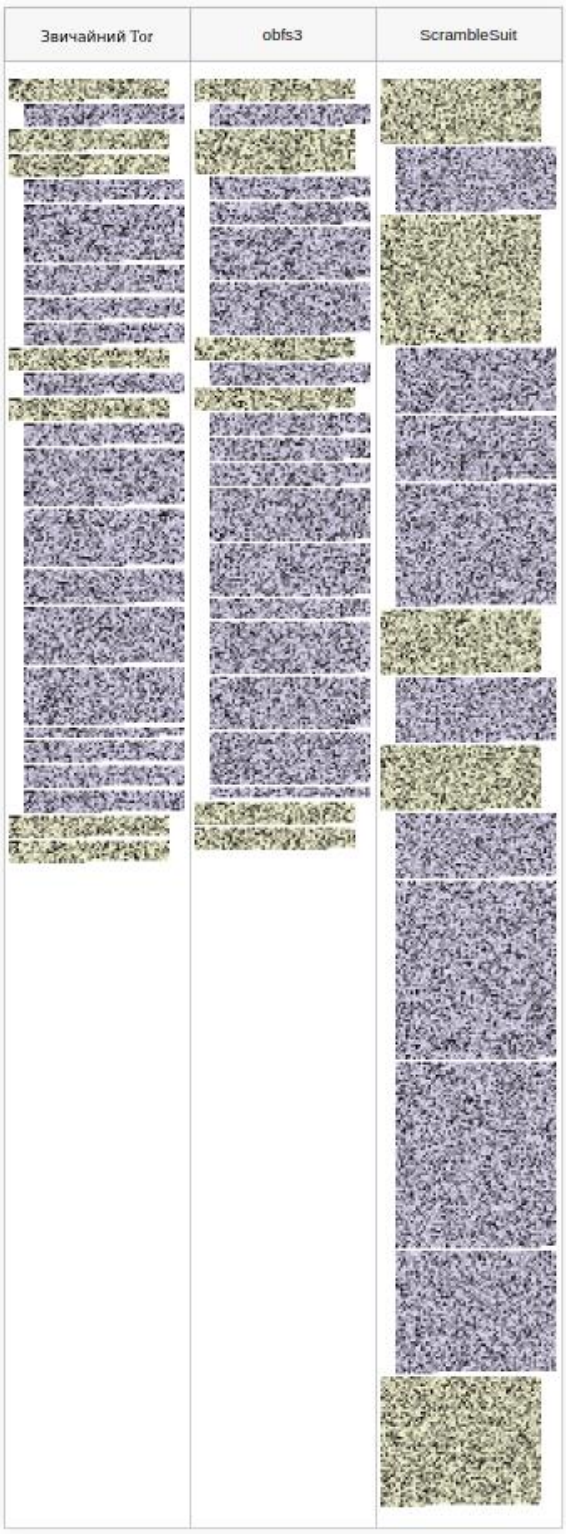

Рисунок 3.20 – Порівняння особливостей трафіку Tor з використанням obfs3 та ScrambleSuit

Як видно з рис. 3.20 obfs3 використовує ті самі розміри пакетів, що і звичайний Tor. ScrambleSuit випадковим чином вибирає кілька розмірів пакетів і виконує заповнення пакетів до цих розмірів, тому загалом надсилається більше байтів.

Крім obfs2, obfs3 та ScrambleSuit існують також інші PG. Наприклад, meek, Bananaphone, що описані в наступних джерелах [35].

Зміна пропускної здатності каналу при підключенні до OpenVPN-сервера із застосування програмного засобу (ПЗ) obfs4proxy-openvpn зображена на рис. 3.21 (використовується динамічний порт).

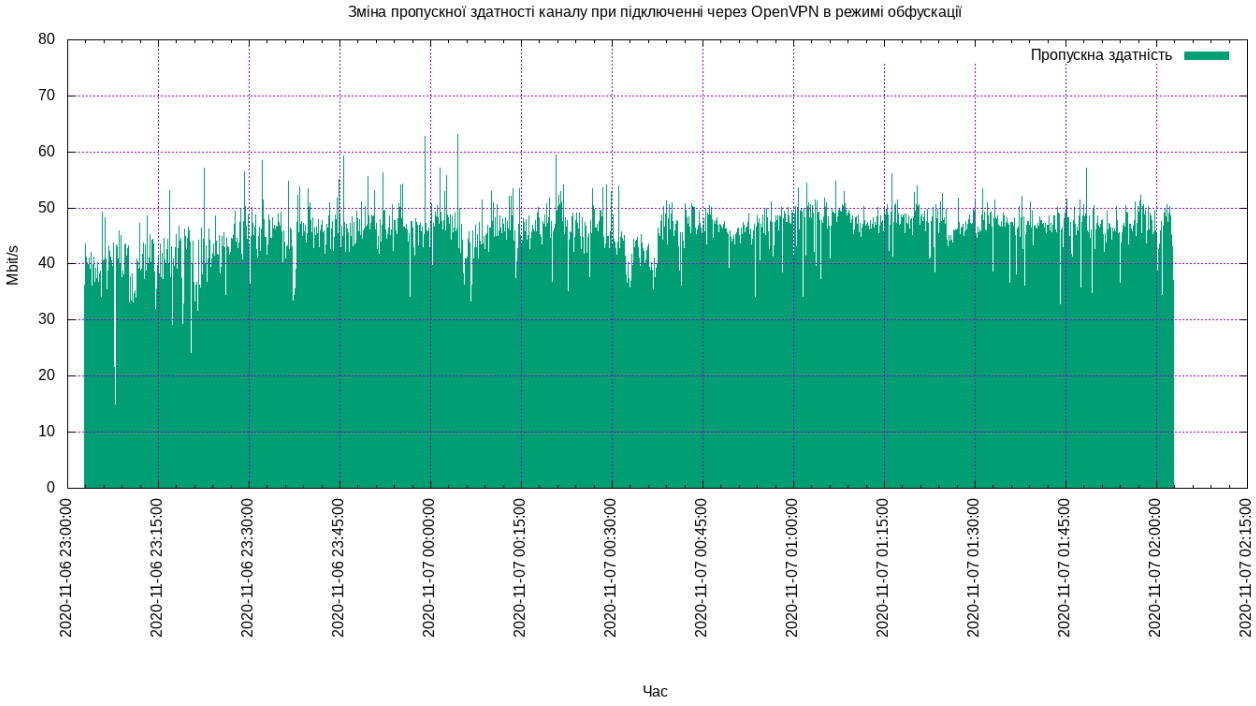

Рисунок 3.21 – Зміна пропускної здатності каналу мережі OpenVPN із застосування програми obfs4proxy-openvpn протягом 3 годин

В даному випадку середня швидкість становила вже 38,9 Мбіт/с (майже на 13 Мбіт/с повільніше, ніж при звичайному підключенні OpenVPN). Кількість «перевідправлень» збільшилася до 343 311 (більш, ніж в чотири рази), що може пояснюватися існуванням додаткового ПЗ (obfs4-proxy), яке вносить додатку затримку, і критично позначається у випадку «TCP над TCP».

3.3.2 Програма забезпечення зміни коду вихідної програми

Stunnel це програма, яка функціонує як SSL-клієнт та (або) SSH-сервер, щоб забезпечити зашифрованим з'єднанням роботу прозорого протоколу TCP без зміни коду вихідної програми. Stunnel може працювати в фоновому режимі або як служба inetd. Stunnel ліцензований загальною публічною ліцензією GNU, що робить вихідний код програми доступним для кожного.

В даний час Stunnel доступний як для Unix, так і Windows платформ. Під час налаштування серверу (або клієнту) Stunnel повинна бути встановлена криптографічна бібліотека OpenSSL тому, що Stunnel не містить криптографічних алгоритмів.

Існують причини використання Stunnel для додавання захисту протоколу, а не для додавання можливостей SSL у програму. Однією з основних причин для використання Stunnel є те, що вихідний код додатку може не бути доступний, особливо якщо це комерційне застосування і розробник вирішує не додавати SSL до програми. Інша причина полягає в тому, що для додавання можливостей SSL до власних програм потрібен час. Stunnel може бути корисним як «запобіжний захід», поки SSL впроваджується та тестується в програмі.

Для демонстрації роботи Stunnel нижче приводиться рис. 3.22.

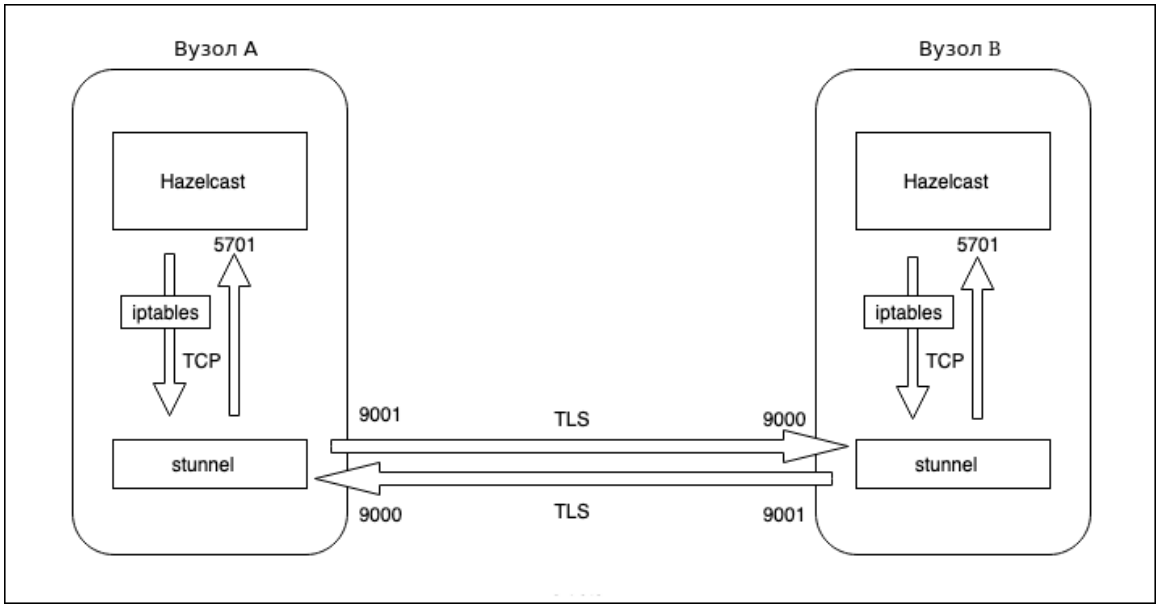

Рисунок 3.22 – Принцип роботи програми Stunnel

На рисунку зображені два вузли. Для прикладу, в якості додатку, до якого додаються можливості SSL/TLS, приведений Hazelcast. В обох сторін встановлений Stunnel, що, по суті, представляє собою програму-посередника. Для підключення додатку через Stunnel додається правило в iptables, що дозволяє перенаправляти трафік додатка не до віддаленного вузла (напряму), а через Stunnel, що встановлений на на мережевому інтерфейсі із зворотним зв'язком (localhost). Так TCP-трафік, додатка, що працює на 5701 порті (зарегестрований порт для Hazelcast), перенаправляється через Stunnel, який у свою чергу використовує порти 9001 (на стороні клієнту) та 9000 (на стороні серверу) для з'єднання з (відповідно) віддаленим вузлом через мережу Інтернет на основі SSL/TLS-з'єднання.

На стороні сервера Stunnel найкраще мати існуючий прозорий текстовий протокол, що працює на localhost замість звичайного мережевого інтерфейсу IP. Існує значне покращення продуктивності роботи мережевої служби через доменний сокет Unix (UDS). На відміну від традиційного мережевого інтерфейсу IP, UDS не повинен працювати з протоколами виправлення помилок. Продуктивність, насправді, може подвоїтися при використанні UDS порівняно з IP, залежно від того, наскільки добре була реалізований механізм UDS [36].

Важливо зазначити, що Stunnel не працюватиме з жодним додатком, заснованим на UDP, такими як DNS, syslog та протоколами потокової передачі мультимедіа. SSL/TLS в основному був розроблений для роботи з протоколом TCP. Окрім функціонування Stunnel, протокол на основі TCP повинен відповідати ще трьом іншим вимогам:

- він не повинен використовувати кілька з'єднань, як, наприклад, є у протоколі FTP (порти управління та даних);
- він не повинен залежати від даних поза межами діапазону;
- віддалений вузол не може використовувати специфічний для програми протокол, такий як ssltelnet, де SSL є домовленим варіантом» [37].

На рис. 3.23 показаний дамп трафіку OpenVPN, що виглядає як звичайний потік TCP пакетів, а на рис. 3.24 трафік OpenVPN при використанні stunnel.

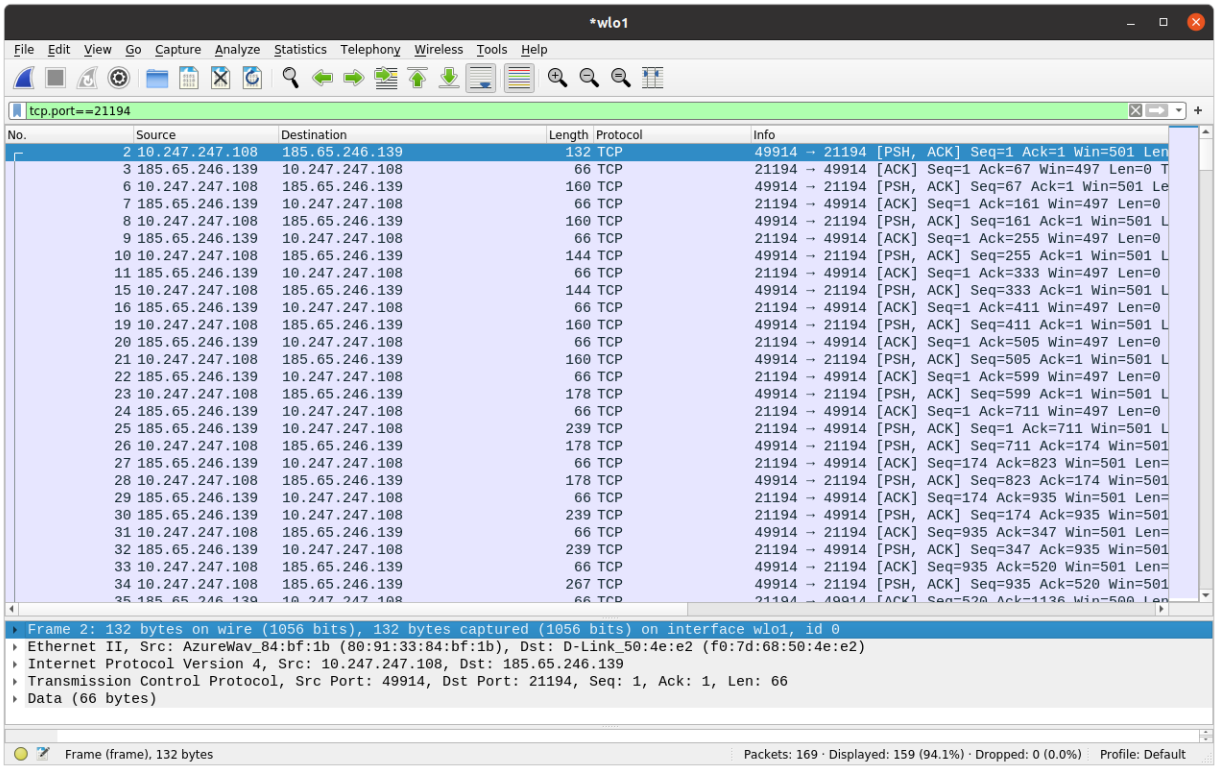

Рисунок 3.23 – Трафік OpenVPN в Wireshark

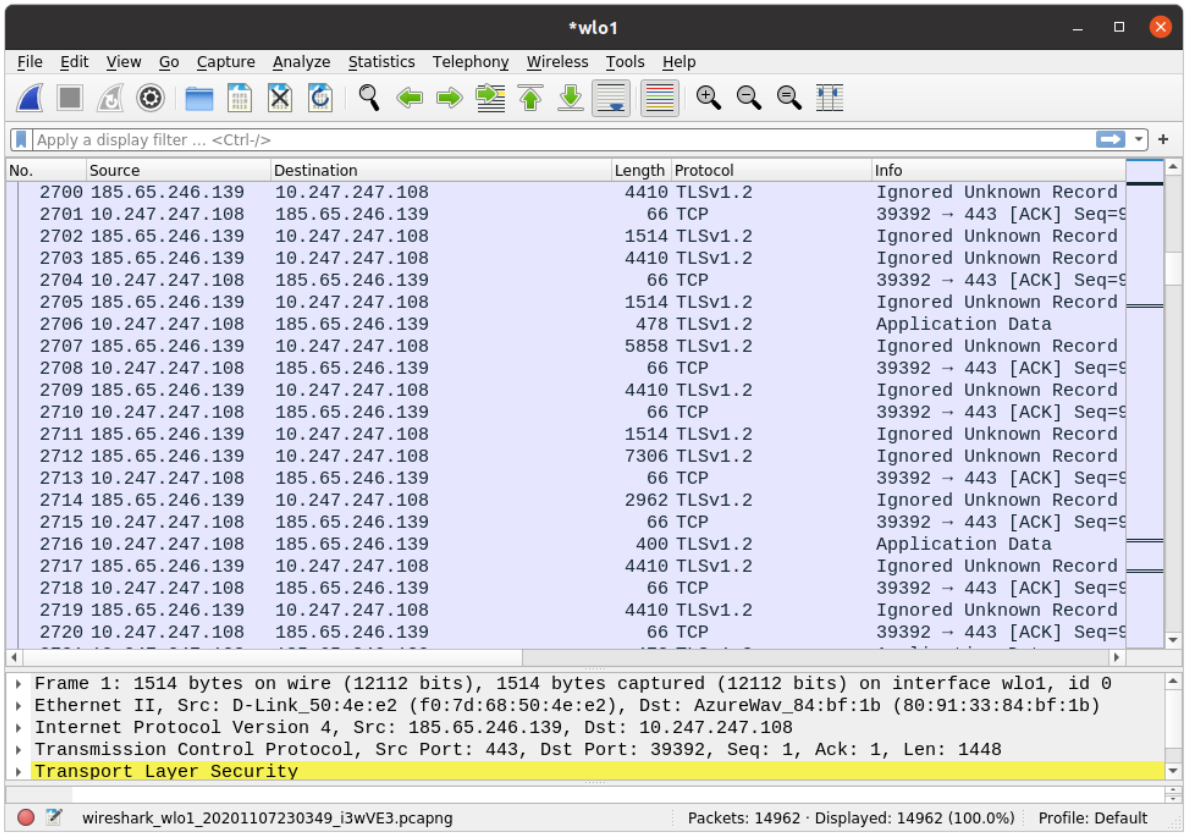

Рисунок 3.24 – Трафік OpenVPN з використанням Stunnel в Wireshark

Існує кілька важливих питань, які слід враховувати при впровадженні служб Stunnel. Дана програма може вимагати більше системних ресурсів, ніж запуск власної програми з підтримкою SSL. Коли Stunnel підключається до додатку та пересилає запити, з’являються принаймні два процеси для обробки запитів, оскільки більшість демонів Unix (sendmail, imap) та всі служби, що базуються на inetd, породжують нові процеси для кожного нового підключення.

Процес Stunnel вважається другим процесом. Налаштування ядра операційної системи – це один із способів вирішити проблему обробки великих таблиць процесів. Крім того, може знадобитися додати більше оперативної та віртуальної пам'яті для обробки додаткових процесів.

Також продуктивність Stunnel може буде покращена шляхом його запуску в фоновому режимі (демону) замість inetd. Так не потрібна ініціалізація SSL при кожному підключенні, оскільки відбувається кешування SSL-сеансу [38].

Ще одне питання, яке слід розглянути перед впровадженням Stunnel, – це потужність системи. Якщо кількість SSL-з'єднань в одиницю часу є великим, це може завантажити систему, оскільки SSL вимагає додаткового процесора обчислювальної потужності для виконання шифрування і дешифрування трафіку. Однією з можливостей є використання плати прискорювача SSL або приладу, який розвантажує криптографічну обробку від основного процесора. Набір інструментів OpenSSL містить можливість зв’язку з акселератором, тому немає необхідності в зміні програми Stunnel .

Дуже важливим питанням є безпека секретного ключа, так як Stunnel не може генерувати зашифровані секретні ключі. Оскільки важливо, забезпечити безпеку систему, яка має закритий ключ і обмежити число людей , які можуть отримати доступ до системи, для підтримки Stunnel використовується бібліотека OpenSSL.

На рис. 3.25 зображено зміну пропускної здатності каналу при підключенні до OpenVPN-сервера з маскування трафіку під HTTPS за допомогою програми Stunnel.

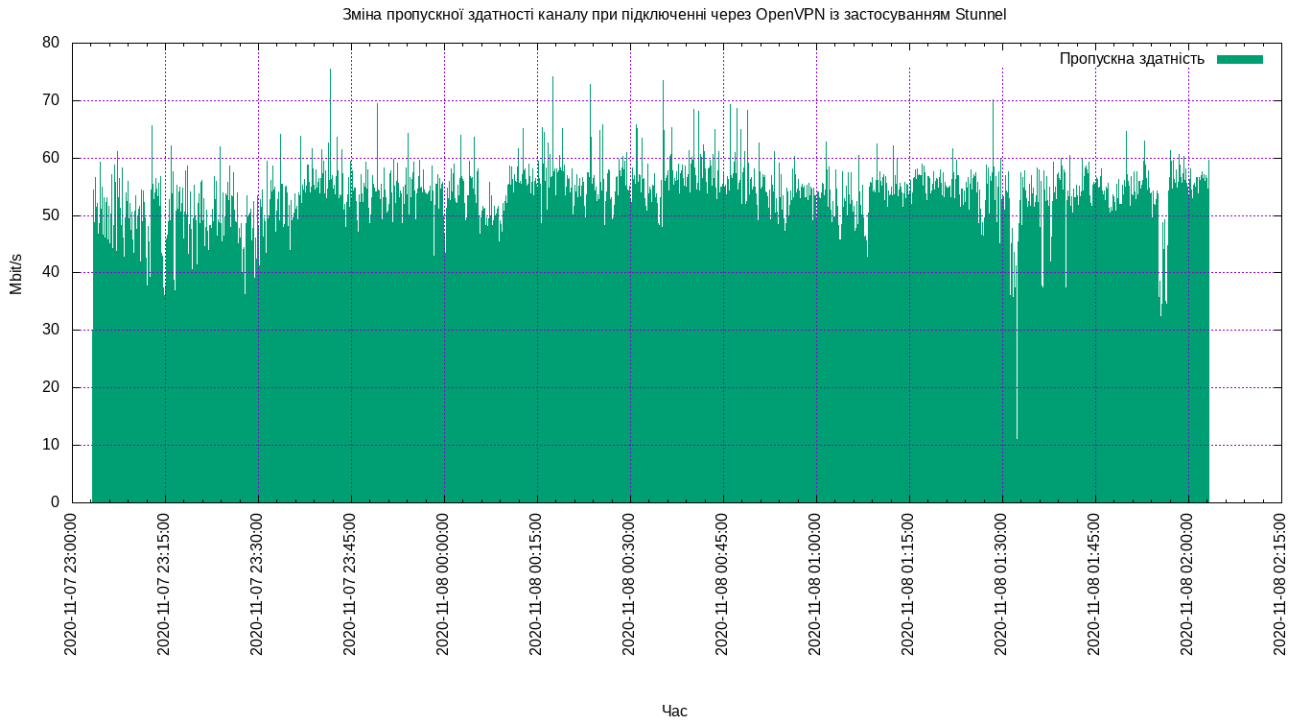

Рисунок 3.25 – Зміна пропускної здатності каналу мережі OpenVPN із застосування програми Stunnel

При застосуванні Stunnel середнє значення пропускної здатності каналу мережі OpenVPN становить 45,1 Мбіт/с. Кількість втрачених TCP-сегментів дорівнює 267 906. На рис. 3.26 можна спостерігати характерні короткочасні сплески трафіку на всьому проміжку вимірювання. Такий трафік можна порівняти з трафіком, що спостерігався раніше, при підключенні через OpenConnect в режимі TLS (рис. 3.9). Обидва рішення були засновані на роботі протоколу SSL/TLS, що працював на 443 порту (HTTPS).

3.4 Висновки до третього розділу

Під час проведення дослідження із різними методами обходу DPI вдалося виявити характерні відмінності трафіку. Слід зазначити, що ні одного методу на момент дослідження заблоковано не було. Також не мали місця бути розриви з’єднання. Після проведення дослідження можна з впевненістю сказати, що політика щодо перенесення трафіку користувачів здійснюється провайдером на основі номеру порту додатку. В цьому можна було переконатися на початку дослідження, наприклад, порівнявши рис. 3.5 та 3.6. Проте не можна сказати

напевно чи аналізуються дані пакету.

Для цього було проведено додаткове дослідження, результати якого приведені у висновках.

Порівняння даних, що були отримані в процесі дослідження методів обходу систем DPI, приводиться в табл. 1.

Таблиця 1 – Порівняння значень параметрів з'єднання методів обходу DPI

	SSH-тунелювання	OpenConnect		Обфускація OpenVPN	
		Режим TLS	Режим DTLS	obfs4	stunnel
Пропускна здатність, Мбіс/с	45,6	39,3	40,3	38,9	45,1
Кількість втрачених сегментів TCP	5 258	87 062		343 311	267 906

Стабільним трафіком можна назвати лише трафік, що проходив через SSH-тунель та працював на стандартному порті 22 із середньою швидкістю 45,6 Мбіт/с (звичайний OpenVPN не може слугувати рішенням для обходу DPI, хоч його трафік також протікає стабільно).

OpenConnect, що був вибраний в якості прикладу VPN-SSL, був протестований у двох режимах: з DTLS та без нього. Хоч DTLS й призначений для прискорення передачі даних, ми можемо спостерігати приріст швидкості лише в 1 Мбіт/с. Така несуттєва перевага може бути пояснена тим, що при вимірюванні пропускної здатності використовувалося TCP-з'єднання. Передача даних за допомогою датаграм DTLS може бути виправдана лише при передачі UDP-пакетів. В обох режимах кількість втрачених сегментів становила до 85 000. Для порівняння, при тестуванні звичайного тунелю OpenVPN така кількість становила 78 000.

Тестування обфускації трафіку OpenVPN дало неоднозначні результати. Середня швидкість при застосуванні програми stunnel становила 45 Мбіт/с, а при передачі трафіку за допомогою obfs4 помітно менша – 38,9 Мбіт/с. В обох методах

кількість «перевідправлень» за 3 години вимірювання становила кілька сотень тисяч сегментів. Таку велику кількість втрачених пакетів можна пояснити ефектом роботи «TCP над TCP» з накладанням на неї додаткової затримки (додатковий час на обфускацію VPN-трафіка).

Слід зазначити, що кожен з приведених методів обходу DPI-аналізу може бути запросто виявленим шляхом застосування активного зондування, і таким чином бути заблокованим. Предметом наступного розділу буде приведення розробки програми для приховання роботи приведених методів обходу DPI від активного зондування.

4 РОЗРОБКА ПРОГРАМИ, СТІЙКОЇ ДО АКТИВНОГО ЗОНДУВАННЯ

Для приховання факту використання методу обходу DPI систем від активного зондування є наразі лише один спосіб: фронтинг домену (що був детально описаний в 2 розділі). Його застосування полягає в наступному:

- розташувати сервер обходу в мережі CDN, блокування якої призведе до значного збитку цензорам в плані інформаційного простору через широке використання CDN багатьма веб-сайтами;
- або приховати сервер обходу за ресурсом, контекст якого не несе в собі ніякої протиправної інформації.

У другому випадку програми обходу, що маскуються під HTTPS (наприклад, OpenConnect або OpenVPN/stunnel) повинні бути приховані за веб-сайтом HTTPS, що використовує стандартний порт 443. Таким чином, одним із компонентів системи фронтингу доменів обов'язково має бути веб-сервер. В даному розділі пропонується розглянути програмні рішення на основі другого способу, вдосконалені мною для застосування фронтингу доменів. В якості веб-серверу мною було вирішено скористатися Nginx.

Nginx позиціонується розробником як простий, швидкий і надійний сервер, не перевантажений функціями [39]. Його застосування доцільно перш за все для статичних веб-сайтів і як зворотного проксі-сервера. Останнє, як буде продемонстровано далі, грає ключову роль у впровадженні фронтингу доменів.

4.1 Програма із застосуванням технології Port Knocking

Технологія Port Knocking як раз призначена для активації «прихованих сервісів» на сервері: це метод зовнішнього «відкриття» порту на брандмауері шляхом створення спроби підключення до порту в певний спосіб. Основною метою Port Knocking є запобігання зловмисникові сканувати систему для потенційно експлуатованих служб. Наприклад, подібним чином часто закривають порт «назовні», на якому працює SSH-демон, щоб уникнути атак типу «брутфорс» [40] і використання 0-day вразливості [40]. В класичному варіанті під Port Knocking розуміють спроби встановлення з'єднання або відправлення пакетів на певні порти хосту в певній послідовності, в результаті чого активується правило брандмауера, що відкриває доступ на певний порт тільки для ініціатора з'єднання.

Якщо відправник не надішле правильну послідовність спроб, захищені порти будуть закритими.

В нашому випадку цей спосіб є не зовсім прийнятним. По-перше, самі «нестандартні» порти можуть бути заблоковані на маршруті від відправника до серверу (система DPI може блокувати всі нестандартні порти), а по-друге якщо ми маскуємо трафік під HTTPS, то і «стукати» потрібно по HTTPS (tcp 443, а не, наприклад, послідовність: tcp 6000, tcp 7000, udp 8000). Тому вирішено було розробити власне рішення на базі Port Knocking, написаний на мові програмування Python. Програмний код приведений в додатку Б.

На рис. 4.0 приведена схема роботи фронтингу домену із застосуванням механізму Port Knocking.

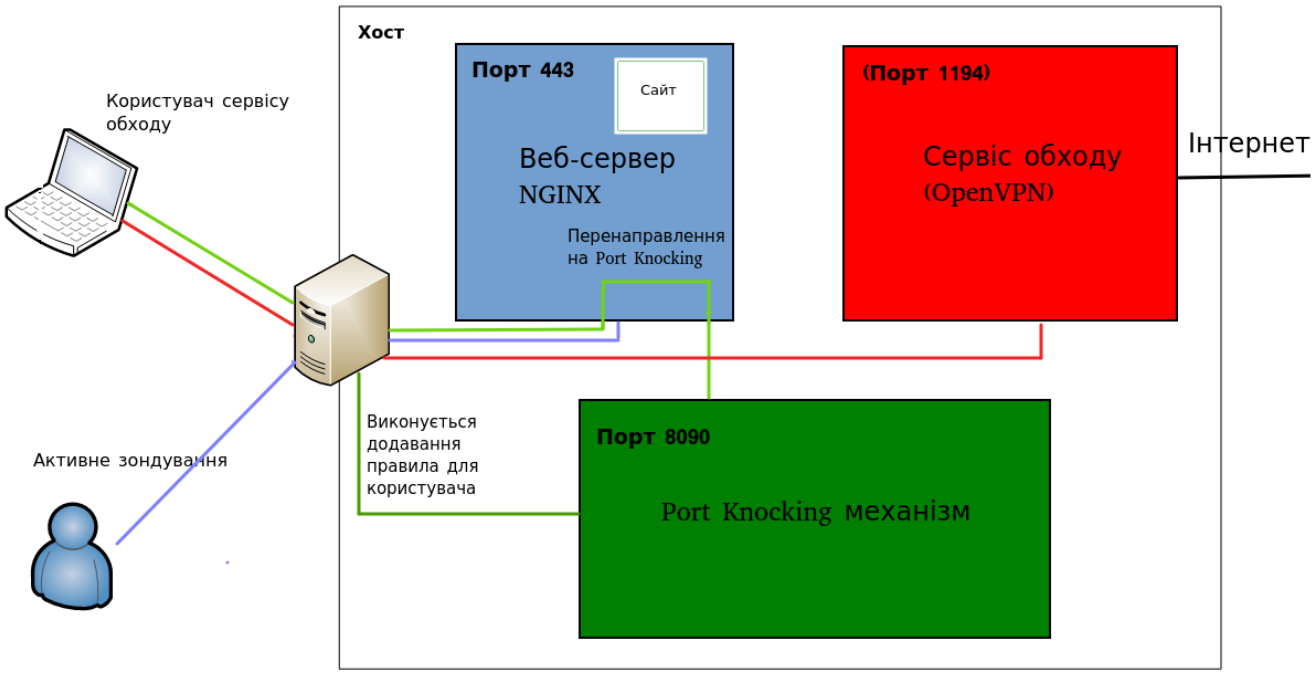

Рисунок 4.1 – Схема роботи фронтингу домену з використанням Port Knocking

Веб-сервер Nginx працює на 443 порту з коректно налаштованими сертифікатами і видає контент (рис. 4.0), що не порушує правила встановлені урядом.

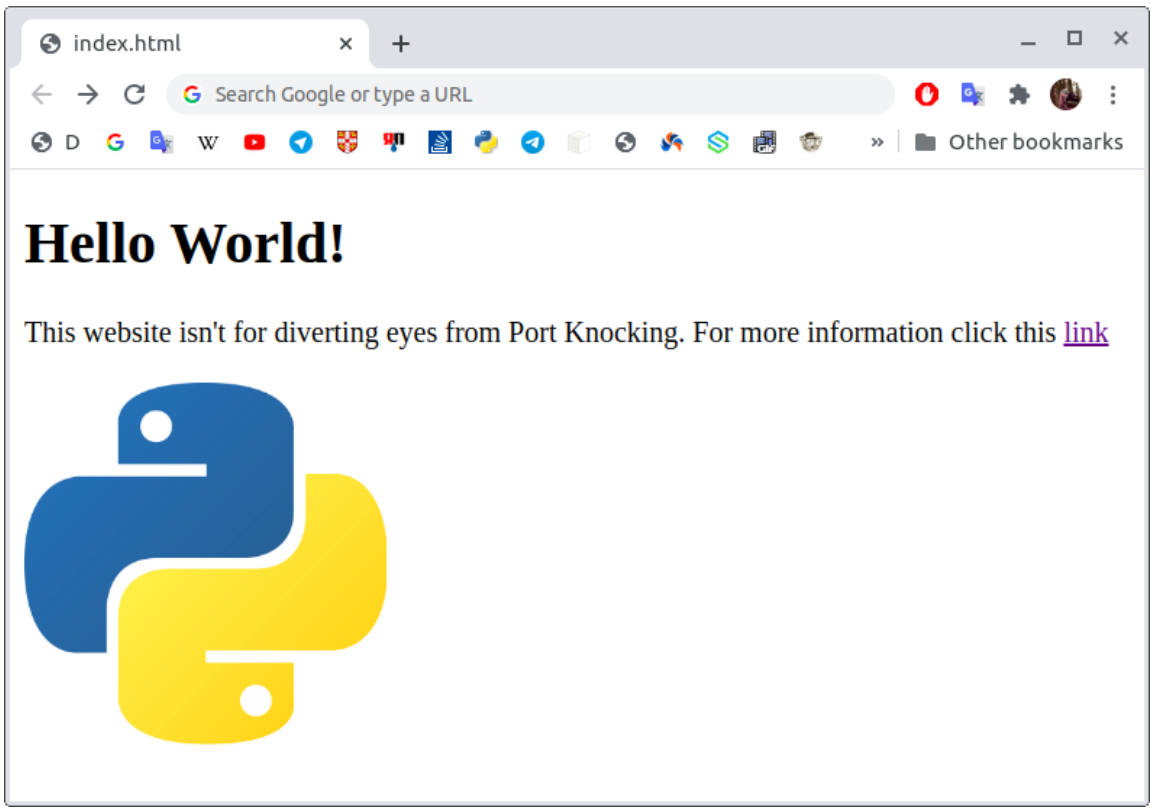

Рисунок 4.0 – Головна веб-сторінка, що налаштована на сервері Nginx

При цьому в конфігураційному файлі Nginx (в додатку В приведена повна конфігурація серверу) містяться рядки виду:

```
location ~ ^/secret/(.*) {
  auth_basic «Administrator Login»;
  auth_basic_user_file /var/www/.htpasswd;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_pass http://127.0.0.1:8080;
}
```

Тепер коли потрібно підключитися до прихованого сервісу (наприклад, OpenVPN), необхідно перейти в браузері, або зробити запит утилітою curl, по певній адресі. Потім пройти стандартну авторизацію, після чого Nginx направить запит до програми Port Knocking (що працює на 8080 порті), яка в свою чергу виконає команду на відкриття доступу до прихованого сервісу для $remote_addr (наприклад, $iptables -t nat -I PREROUTING -p tcp -s $remote_addr -dport 443 -j REDIRECT --to-port 1194). Після цього «підняти» тунель сервісу (наприклад, OpenVPN).

Можна сказати, що користувач таким чином проходить двоетапну

автентифікацію: для активації сервісу він повинен спочатку ввести URL-адресу із певним секретом («secret» в приведеній вище URL), а потім ввести логін та пароль.

Також можна привести певний запит для команди, що відміняє встановлене правило брандмауера. Після чого встановлене з’єднання залишається активним, але підключитися до сервісу без повторної процедури Port Knocking вже не вийде.

Таким чином можна сконфігурувати навіть декілька різних сервісів, в тому числі, використовуючи IPv6 (адреси типу v6 в заголовку X-Real-IP також підтримуються).

4.2 Програма із застосування протоколу WebSocket

В минулому рішенні використовувалося два з’єднання: перший – це запит, для встановлення правилу брандмауера і друге – це, власне, встановлення тунелю із сервером обходу. Річ у тім, що два потоки HTTPS можуть відрізнятися, адже сервіси використовуються різні (веб-сервер Nginx та VPN-SSL). Особливо різниця може бути помічена при TLS-рукостисканні, де зазначені алгоритми, сертифікати тощо, або при аналізі розмірів пакетів. Так з’являється додатковий привід для підозри з боку операторів DPI. Тому крайнє необхідно дотримуватися загальних правил для двох потоків. Так у минулому рішенні використовувалися однакові сертифікати як для веб-серверу та OpenVPN серверу. Однак незважаючи на це існує великий ризик бути викритими просунутими DPI-системами. Тому кращим рішення слугувало б встановлення єдиного з’єднання через веб-сервер до сервісу обходу, та з передачею через нього даних (щось накшталт HTTPS-тунелювання, тобто передачі будь-яких даних через HTTPS, а не лише гіпертексту). В результату аналізу можливих рішень я зупинився на протоколі WebSocket.

WebSocket – протокол зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між браузером і веб-сервером в режимі реального часу. Він також може бути використаний для будь-якого клієнтського або серверного додатка. Протокол WebSocket – це незалежний протокол, заснований на протоколі TCP. Він робить можливим більш тісний контакт між браузером і веб-сайтом, сприяючи поширенню інтерактивного вмісту та створення додатків реального часу. Для встановлення з'єднання WebSocket клієнт і сервер використовують протокол, схожий на HTTP. Клієнт формує особливий HTTP-запит, на який сервер відповідає певним чином. Відразу після відправки відповіді WebSocket-з'єднання

вважається встановленим, клієнт і сервер можуть починати двонаправлений обмін повідомленнями з цього ж TCP-з'єднання.

На рис. 4.0 для наочності порівнюються принципи обміну даними в Websocket та HTTP.

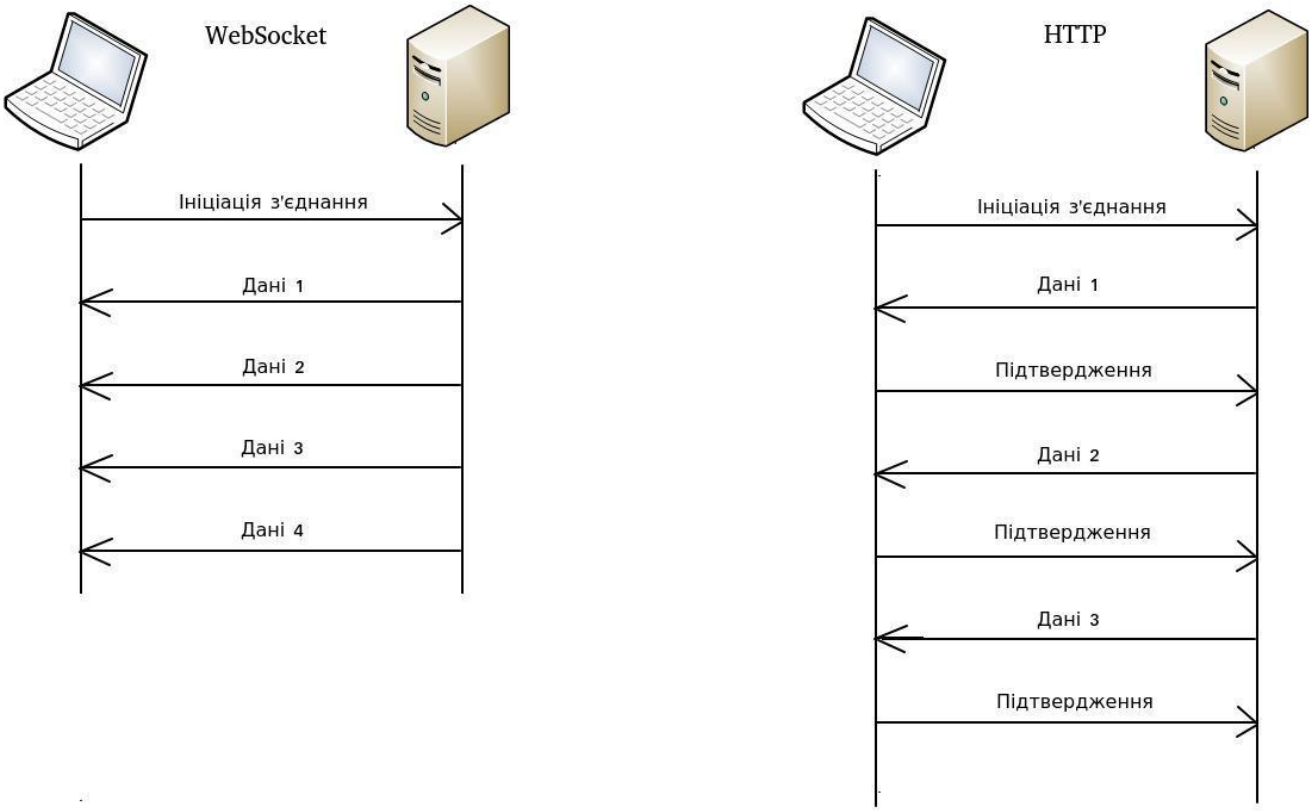

Рисунок 4.3 – Обмін даними в протоколах Websocket та HTTP

Для створення тунелю Websocket, через який будуть передаватися дані, в даній роботі використовується програма wstunnel написана на JavaScript [41]. Клієнт формує особливий HTTP-запит (з урахуванням переходу потім на WebSocket), на який сервер відповідає згодою, і після невеликого рукостискання може починатися передача даних по тому ж TCP-з'єднанню. Зі сторони, відповідно, трафік виглядає точнісінько як звичайний HTTPS-трафік і не потребує встановлення ніяких додаткових з'єднань. Поверх тунелю можно підключитися до звичайного SOCKS-проксі (наприклад, Dante), або OpenVPN (хоча це може бути навіть надміру).

На рис. 4.4 приведена схема роботи фронтингу доменів з використанням програми wstunnel для подальшого направлення трафіку користувача через проксі.

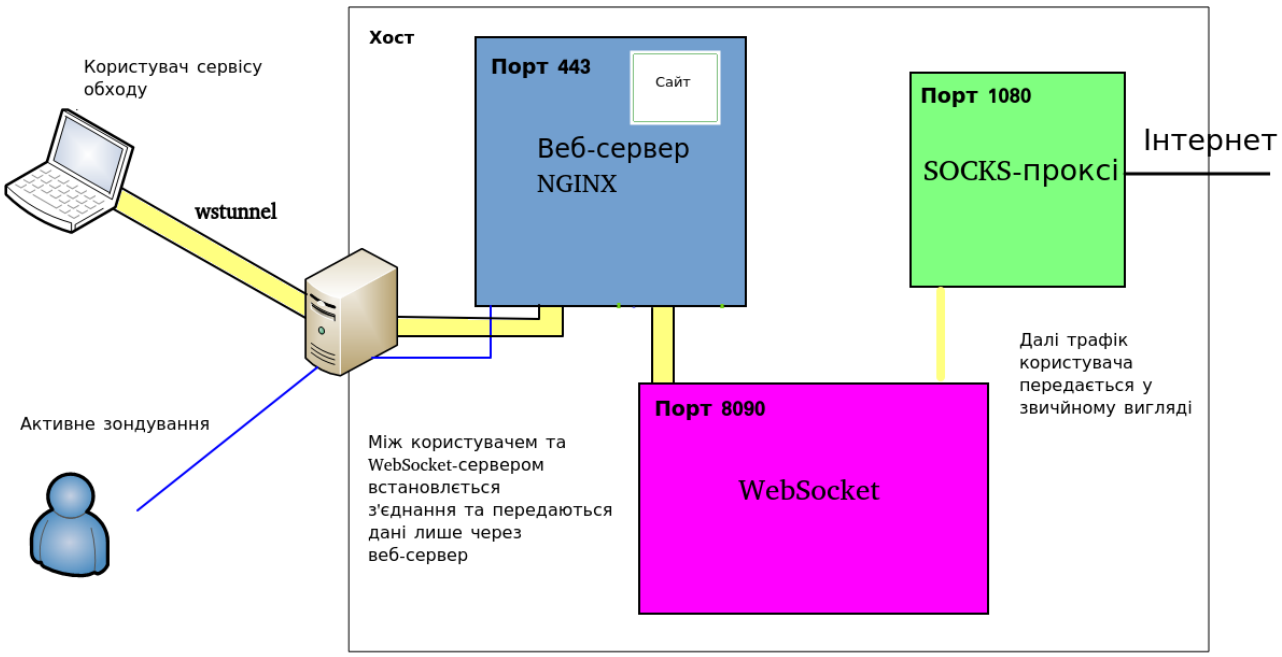

Рисунок 4.4 – Схема роботи фронтингу доменів з використанням програми wstunnel

Конфігурація веб-серверу Nginx для перенаправлення трафіку користувача до WebSocket-серверу зводиться до написання наступних рядків (додаток В):

```
location /secret {
  proxy_pass http://127.0.0.1:8090;
  proxy_http_version 1.1;
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection «Upgrade»;
}
```

На сервері налаштовуємо WebSocket «слухати» порт 8090, і перенаправляємо трафік через проксі SOCKS5, що працює на порті 1080, наступною командою:

```
$wstunnel -s 127.0.0.1:8090 -t 127.0.0.1:1080
```

На стороні клієнту «піднімаємо» WebSocket-тунель командою:

```
$wstunnel -t 6666 wss://<IP-адреса серверу>/secret/ws
```

Таким чином на клієнтському хості програмою wstunnel створюється SOCKS-проксі, що «слухає» порт 6666, і далі направляє весь трафік через тунель wstunnel.

Для прикладу, налаштування браузерів для роботи через SOCKS-проксі на ОС Ubuntu показано на рис. 4.5.

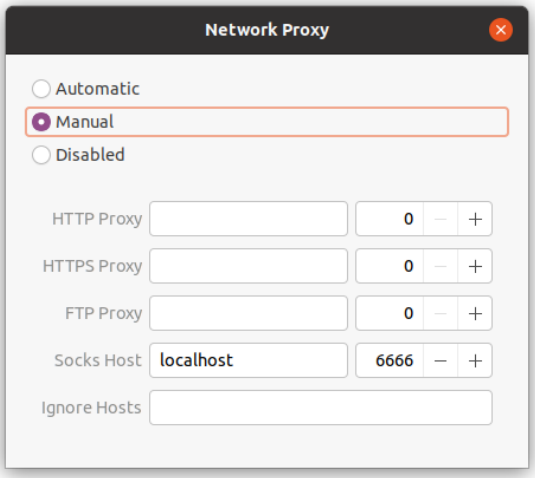

Рисунок 4.4 – Приклад налаштування роботи додатку (веб-браузеру) через проксі-сервер SOCKS

Тепер увесь веб-трафік буде направлений через тунель. Його аналіз в програмі Wireshark приведений на рис. 4.5.

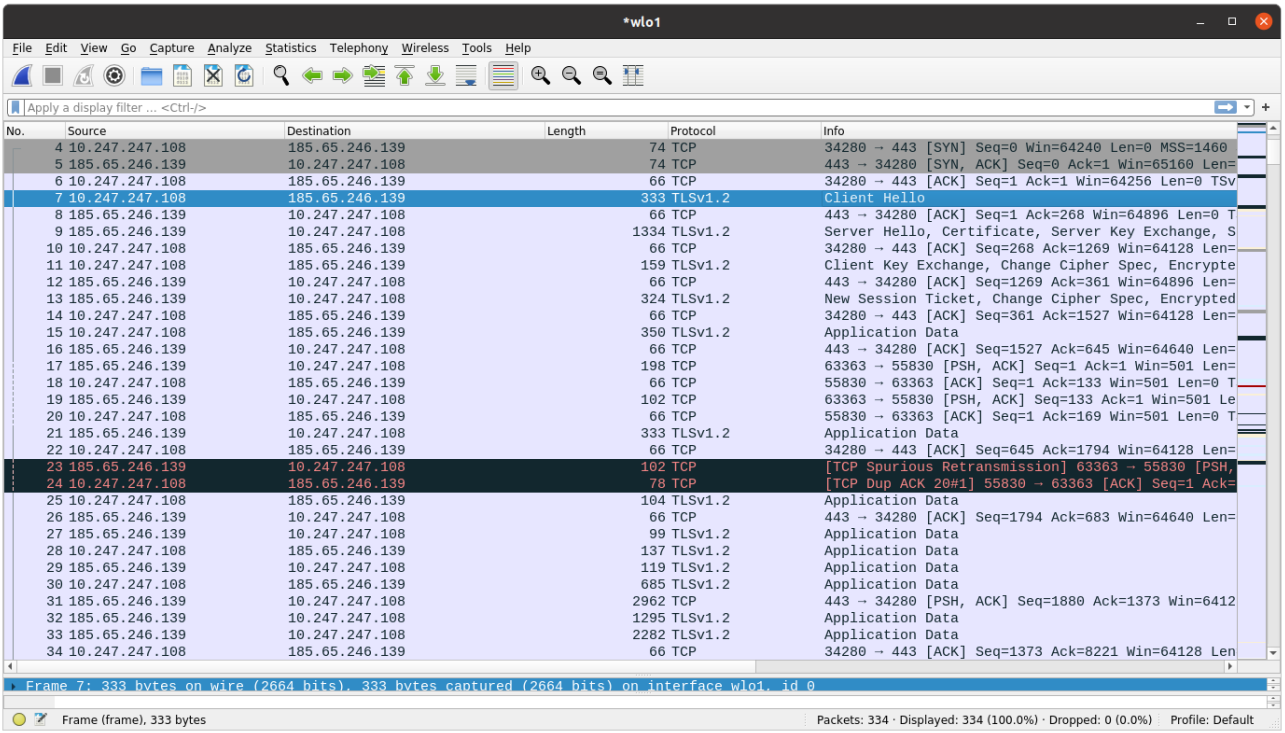

Рисунок 4.5 – Аналіз веб-трафіку, що проходить через тунель wstunnel

У першій та другій спробі повноцінне вимірювання пропускної здатності

каналу при підключенні через тунель wstunnel зазнало невдачі. Так у певний момент часу швидкість передачі даних падає до нуля, а згодом робота CPU досягає 100%, що може критично вплинути на технічний стан комп'ютеру (тому тестування було перервано). Під час роботи над дипломом схоже вже траплялося в процесі дослідження програм обходу DPI, тому це не може бути пояснено виною програми wstunnel. Скоріше за все це недолік утиліти iperf3, що виявляє себе в збільшенні навантаження на процесор з невідомих мені причин.

На рис. 4.6 приводиться візуалізація даних вимірювання пропускної здатності каналу через підключення по тунелю wstunnel, що були отримані з другої спроби. Вимірювання тривало до години часу.

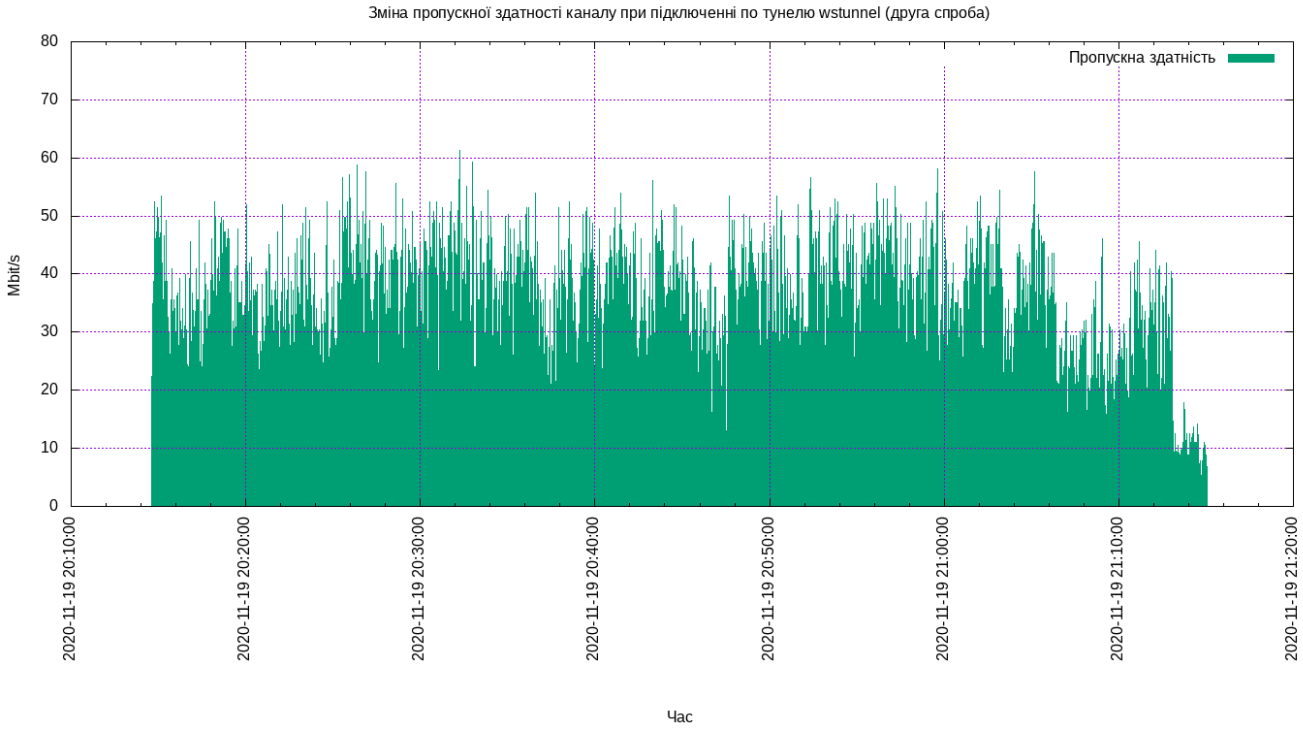

Рисунок 4.5 – Зміна пропускної здатності каналу через підключення по тунелю wstunnel

В кінці графіку (рис. 4.5) з певного моменту часу (за 5 хвилин до «розриву» тестування) можна спостерігати різкий спад швидкості до 10 Мбіт/с. Проте це не може бути передвісником про раптове закінчення тестування, адже в першій спробі такого не спостерігається.

Трафік досить нестабільний, а середня швидкість передачі становить лише 28,2 Мбіт/с, що можна б було пояснити проведенням вимірювання у вечірній, а не в нічний час (як було у випадку всіх попередніх дослідженнях), проте в першій

спробі, коли початок часу вимірювання був 23:00, характер трафіку через 15 хвилин став схожим на той, що можна спостерігати при другій спробі. Ймовірно, що в такий спосіб виконується політика провайдеру до HTTPS-трафіку при високих навантаженнях на мережу іншими протоколами/додатками.

За час вимірювання було втрачено 2678 TCP-сегментів, що є нормальним показником.

4.3 Висновки до четвертого розділу

В якості рішень, стійких до активного зондування, в даному розділі приводяться дві програми, основна робота яких це приховання сервісу обходу за веб-сайтом.

Програма з використанням механізму Port Knocking, не вимагає встановленню додаткового програмного забезпечення (ПО), а зводиться лише до написанню невеликої кількості коду (це стосується, зокрема, ОС, що вже мають інтерпретатор Python, наприклад, Ubuntu), проте вимагає наявності будь-якого методу обходу.

У випадку тунелювання трафіку через WebSocket теж при необхідності можна встановити з’єднання із програмою обходу. Але такий спосіб має певну збитковість (пропускання тунелю VPN через тунель wstunnel), що може впливати на ефективність передачі трафіку мережею. Однак при веб-серфінгу така програма гарно підходить, адже можна використовувати тунелювання трафіку через звичайний проксі сервер. Слід також зазначити, що дане рішення крім встановлення WebSocket-серверу вимагає відповідно наявності WebSocket-клієнта.

ВИСНОВКИ

В атестаційній роботі проаналізовано рівні перевірки пакетів, а саме: поверхневої перевірки пакетів (SPI), середнього рівня перевірки (MPI) та глибокої перевірки пакетів (DPI), вказано на їх роль у застосуванні в інформаційних мережах. Приводилися переваги та недоліки DPI відносно загальноприйнятих методів аналізу заголовків мережевих пакетів в інформаційних мережах. Також аналізувався технічний потенціал DPI: його вплив на економічну та політичну складові сучасного світу.

Проаналізовано роботу активного зондування, як допоміжного механізму DPI-систем. Досліджено принципи роботи методів боротьби з активним зондуванням: метод загального обходу, заснованого на HTTPS, систему обходу Snowflake. Приведено необхідні умови для приховання факту використання обходу DPI-систем для розробки програми, стійкої до активного зондування.

Основною частиною роботи виступав аналіз відомих програм, стійких до систем DPI. Для дослідження програм не лише якісно, а й у кількісному відношенні проводилися вимірювання, на основі яких робилися висновки щодо тієї чи іншої програми обходу DPI. Так вдалось встановити, що провайдер, що пропускав трафік між клієнтом та сервером, використовував інформацію в транспортних заголовках (порт, на якому працював додаток).

Виявлено характерні відмінності трафіку. Проаналізовано, що стабільним трафіком можна назвати лише трафік, що проходив через SSH-тунель та працював на стандартному порті 22 із середньою швидкістю 45,6 Мбіт/с. Протестовано трафік OpenConnect у двох режимах: TLS та DTLS. Проведено тестування обфускації трафіку OpenVPN.

Для встановлення факту використання провайдером системи DPI було проведено додаткове дослідження (додаток Г), однак підтвердження факту використання провайдером системи DPI, як і його спростування, не може бути зроблено через неявне проведення політики провайдера щодо трафіку, що пропускався через його мережу під час дослідження.

Не можна й вибрати серед представлених програм кращу, адже робота кожної залежить від розвитку DPI-системи та політики цензурування. Висока продуктивність у цьому випадку грає другорядну роль.

Також були приведені програми, що можуть бути використані для приховання факту використання сервісу обходу від активного зондування. Так приводилася програма власної розробки, що може бути розглянута як один з засобів проти активного зондування, і може слугувати об’єктом для подальших розробок. Було організовано тунель на базі протоколу WebSocket для застосування фронтингу доменів. Приведено переваги та недоліки представлених програм, стійких до активного зондування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Hafner K. Where Wizards Stay Up Late: The Origins of the Internet / K. Hafner, M. Lyon. – New York: Simon & Schuster, 1999. – 304 с.
2. Zalewski M. Silence on the Wire: a Field Guide to Passive Reconnaisance and Indirect Attacks / Zalewski. – San Francisco: No Starch Press, 2005. – 174 с.
3. Abbate J. Inventing the Internet / Abbate. – Cambridge, Mass.: The MIT Press, 1999. – 177 с.
4. Gune D. Parsing Techniques: A Practical Guide / D. Gune, C. Jacobs. – West Sussex: Ellis Horwood Limited, 2008. – 164 с.
5. Sourdis I. Designs & Algorithms for Packet and Content Inspection / Ioannis Sourdis. – Delft: Delft: TU, 2007. – 185 с.
6. Mochalski K. Copyright Protection in the Internet [Електронний ресурс] / K. Mochalski, H. Schulze, F. Stummer // Whitepaper). – 2103.
7. Chester J. The End of the Internet? [Електронний ресурс] / Chester // The Nation. – 2006.
8. Finnie G. Report) ISP Traffic Management Technologies: The State of the Art [Електронний ресурс] / Finnie // 2009. – 2009.
9. Dameon D. Essential Check Point Firewall-1 NG(TM): An Installation, Configuration, and Troubleshooting Guide [Електронний ресурс] / Dameon // Addison-Wesley Professional. – 2004.
10. Porter T. The Perils of Deep Packet Inspection [Електронний ресурс] / T. Porter // Symantic Corporation. – 2010.
11. Anderson N. Deep packet inspection meets net neutralirty [Електронний ресурс] / N. Anderson. // Ars Technica
12. Розкриття трафіку Skype: коли випадковість грає з вами / [Д. Бонфільо, М. Мікела, М. Марко та ін.]. // Огляд комп’ютерних комунікацій. – №37. – с. 37–48.
13. Lerman S. Net Neutrality and Deep Packet Inspection: Discourse and Practice / Lerman. – Ottawa: A Collection of Essays from Industry Experts, 2009.
14. The Great Firewall: China's Web Users Battle Censorship [Електронний ресурс] // Time. – 2010.
15. Iran To Work With China To Create National Internet System [Електронний ресурс] // www.rferl.org. – 2020.

16. Roya D. Learning more about the GFW's active probing system | Tor Blog [Електронний ресурс] / Roya // blog.torproject.org. – 2019.

17. Tor Games [Електронний ресурс] // people.cs.umass.edu. – 2019.

18. Usage Statistics and Market Share of JavaScript Content Delivery Networks for Websites [Електронний ресурс] // W3Techs. – 2019.

19. A Survey of Worldwide Censorship Techniques [Електронний ресурс] // ietf.org. – 2018.

20. Beck U. World Risk Society / U. Beck. – Cambridge, UK: Polity, 2015 – 192 с.

21. Tracking Malware With Public Proxy Lists [Електронний ресурс] // SANS Institute. – 2020.

22. CEO Датагруп: Блокировка части Рунета [Електронний ресурс] // Ліга.Бізнес. – 2017.

23. Kanjilal C. Deep Packet Inspection and How It Blocks VPN Services [Електронний ресурс] / Kanjilal. – 2017.

24. Dimming the Internet: Detecting Throttling as a Mechanism of Censorship in Iran [Електронний ресурс] // Anderson C. – 2013.

25. Тобкін К. CheckPoint наступного покоління за допомогою адміністрування безпеки розвідки додатків / К. Тобкін, Д. Клігерман. – Рокленд, Массачусетс: Syngress Publishing, Inc., 2004.

26. Roberts H. Circumvention Tool Evaluation [Електронний ресурс] / H. Roberts, E. Zuckerman // Tech. rep. – 2011.

27. Why TCP Over TCP Is A Bad Idea [Електронний ресурс] // Olaf Titz. – 2001.

28. Work in China? [Електронний ресурс] // OpenVPN Support Forum. – 2019.

29. MacKinnon R. Consent of the Networked: The Worldwide Struggle for Internet Freedom / MacKinnon. – New York: Basic Books, 2012 – 352 с.

30. 'Father' of China's Great Firewall Shouted Off Own Microblog – China Real Time Report – WSJ [Електронний ресурс] // Wall Street Journal. – 2010.

31. China's Internet censorship: A WTO challenge is long overdue [Електронний ресурс] // TechPolicyDaily – 2016.

32. User Reputation based Tor Bridge Distribution with Privacy Preservation [Електронний ресурс] // www-users.cs. – 2020.

33. Obfuscation: Cloaking your Code from Prying Eyes [Електронний ресурс] // www.devx.com

34. Bridge users by transport. [Електронний ресурс] // Tor Metrics. – 2017.

35. Dunwoody M. Domain Fronting With TOR [Електронний ресурс] / Dunwoody // FireEye Threat Research Blog. – 2017.

36. O'Donovan B. Secure Communication with Stunnel [Електронний ресурс] / 'Donovan // Linux Gazette. – 2004.

37. Сордіус І. Брифінг про програму модернізації перехоплення / І. Сордіус. – Делфт: TU.Delft, 2007.

38. Deprecating Secure Sockets Layer Version 3.0 [Електронний ресурс] // RFC. – 2017.

39. Use NGINX as a Front-end Proxy and Software Load Balancer [Електронний ресурс] // Linode Guides & Tutorials. – 2018.

40. Rash M. Protecting SSH Servers with Single Packet Authorization [Електронний ресурс] / Rash. – 2007.

41. wstunnel [Електронний ресурс]

42. Кодаченко Р. Ю. Огляд процедур голосової аутентифікації / Р. Ю. Кодаченко. // Міжнародна науково-практична конференція “Інфокомунікації – сучасність та майбутнє. – 2019. – №9. – С. 155.

43. Кодаченко Р. Ю. Хмарна АТС на базі системи віртуалізації / Р. Ю. Кодаченко, В. Є. Цилюрік. // Матеріали XXIV міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». – 2020. – №23. – С. 72.

44. Кодаченко Р. Ю. Забезпечення захисту інформації в мережах на канальному рівні за допомогою протоколу Openvpn / Р. Ю. Кодаченко. // Матеріали XXIV міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». – 2020. – №23. – С. 106.