ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: **«Розроблення програмного забезпечення для дослідження протокола TLS (Transport layer Security)»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2020
Мулько А.В.
Керівник: Одарущенко О. Б.
Рецензент: Ковальчук С. Б

Полтава – 2024 року

ВСТУП

В сучасному цифровому світі, де інформаційні технології стають все більш невід'ємною складовою нашого повсякденного життя, питання кібербезпеки стає набагато більш актуальним та важливим. Кіберпростір, як ніколи раніше, є витоком не лише можливостей, а й ризиків. Від кібератак та витоків даних до кібершпигунства та кібертероризму - загрози в інтернеті стають все більш вдосконаленими та складними.

Тема кібербезпеки охоплює широкий спектр аспектів, від технічних до соціальних, від правових до етичних. Її важливість відзначається на всіх рівнях - від індивідуальних користувачів до великих корпорацій та урядових органів. Забезпечення безпеки в кіберпросторі вимагає не лише розробки технологічних рішень та застосування криптографічних методів, але й виховання кіберграмотності серед користувачів, розробки політики кібербезпеки на рівні держав та міжнародних співтовариств, а також співпраці між галузями та країнами [1].

У світлі постійного розвитку технологій та зростаючої кількості кіберзагроз, розуміння та вивчення питань кібербезпеки стає важливим завданням для всіх зацікавлених сторін.

Протокол TLS (Transport Layer Security) є основним засобом забезпечення безпеки комунікацій в Інтернеті. Він забезпечує конфіденційність, цілісність та автентифікацію даних, що передаються через мережу. TLS дозволяє створювати захищені з'єднання між клієнтом і сервером, що відвертає можливість атак перехоплення, підміни та розголошення інформації [1, 2].

Ця тема спрямована на розгляд принципів безпеки, які реалізуються через протокол TLS. Метою її проведення є краще розуміння того, як TLS працює, які криптографічні методи використовуються для захисту даних та як саме цей протокол сприяє забезпеченню безпеки в Інтернеті.

Актуальність теми. Тема має високий рівень актуальності, близько 90-95%. Це через те, що кібербезпека стає все більш важливою у світі, де зростає

кількість кібератак і потенційних загроз для інформаційної безпеки. Таким чином, дослідження та розвиток інструментів для аналізу та вдосконалення протоколу TLS має велике значення для багатьох сфер, включаючи бізнес, урядові організації, а також індивідуальних користувачів Інтернету. Відповідно до цього, інвестування у дослідження та розвиток нових методів кібербезпеки для TLS може сприяти зменшенню ризиків та втрат, пов'язаних із кібератаками, і забезпечити більшу впевненість у безпеці віртуального середовища.

Мета роботи – провести аналіз і розробити програмне забезпечення для дослідження протоколу TLS (Transport layer Security).

Отже, мета роботи полягає в розумінні, аналізі та покращенні протоколу TLS, а також у розробці нових інструментів для забезпечення безпеки в інтернеті.

Предмет дослідження кваліфікаційної роботи – методи і інструментальні засоби розроблення криптографічних протоколів.

Об'єкт дослідження кваліфікаційної роботи – процеси розроблення криптографічних протоколів.

Методи дослідження: системний аналіз (розділи 1, 2), розроблення (розділи 2, 3).

Інформаційна база, що використовувалася: публікації в періодичних виданнях, вебресурси.

Практична значущість: Розроблено та опубліковано програмне забезпечення для дослідження протоколу TLS. Сформовано універсальний робочий підхід до розробки та реалізації програмних забезпечень згідно існуючих рекомендацій для подальшого використання.

Робота складається зі вступу, трьох розділів, висновків та списку літератури. Обсяг роботи становить 49 сторінки, 3 таблиця, 22 рисунка, 1 додаток.

РОЗДІЛ 1
АНАЛІЗ БЕЗПЕКИ ТРАНСПОРТНОГО РІВНЯ TRANSPORT LAYER SECURITY (TLS) ПРОГРАМУВАННЯ TLS

1.1 Безпека інформаційних технологій

Безпека інформаційних технологій – це комплекс заходів, спрямованих на захист інформаційних систем, даних та інфраструктури від несанкціонованого доступу, змін, видалень або збоїв, які можуть загрожувати конфіденційності, цілісності та доступності інформації. Основні аспекти цього заходу включають захист мережі, даних, управління ідентифікацією та доступом. Управління вразливостями включають безпеку додатків, моніторинг та виявлення загроз, а також навчання користувачів щодо правил безпеки [3].

В англійській мові поняття безпеки IT розпадається на дві основні категорії. Поняття функціональної безпеки (англ. safety) означає, що система працює коректно та виконує потрібні функції відповідно до вимог і намірів її власника. З іншого боку, поняття інформаційної безпеки (англ. security) стосується захисту процесу технічної обробки інформації. Така система повинна забезпечувати захист від несанкціонованого доступу до даних та запобігати їхній втраті у разі збоїв.

Інформаційна безпека, в найзагальнішому сенсі, охоплює комплекс заходів, спрямованих на зменшення можливих ризиків та мінімізацію збитків, які може понести підприємство внаслідок розголошення конфіденційної інформації [4]. З цієї перспективи, інформаційна безпека розглядається як економічний параметр, який має бути врахований у стратегії підприємства. Дані можна розглядати як товар або цінність, що підлягає захисту, і тому вони повинні бути доступні лише авторизованим користувачам або програмам.

Основні аспекти безпеки інформаційних технологій включають в себе [5]:

– захист мережі: встановлення і налагодження заходів безпеки на рівні мережі, включаючи мережеві брандмауери, системи виявлення вторгнень (IDS), віртуальні приватні мережі (VPN) та інші технічні засоби;

– захист даних: використання шифрування даних, контроль доступу, аудиту доступу до даних та інші методи для забезпечення конфіденційності та цілісності інформації;

– управління ідентифікацією і доступом: реалізація систем автентифікації, авторизації та адміністрування користувачів та їх доступу до ресурсів;

– захист від вразливостей: проведення регулярного аналізу вразливостей і патчінгу систем для запобігання експлуатації вразливостей зловмисниками;

– безпека додатків: розробка та впровадження заходів безпеки на рівні додатків, включаючи перевірку коду на вразливості та захист від атак на програмне забезпечення;

– моніторинг і виявлення загроз: застосування систем моніторингу та аналізу подій для виявлення аномальної активності та потенційних загроз безпеці;

– навчання та свідомість користувачів: освіта персоналу щодо правил безпеки та прийомів поведінки, що допомагають уникнути загроз безпеці.

Ці заходи спільно допомагають забезпечити надійний рівень захисту інформаційних технологій в організаціях та інших сферах діяльності.

Спосіб несанкціонованого доступу (НСД) – це сукупність методів і процедур, спрямованих на отримання (добування) незаконного доступу до охоронюваної інформації та можливість впливу на неї, таким чином, щоб отримати конкурентну перевагу, змінити інформаційні потоки конкурента на свою користь або завдати шкоди конкурентові, включаючи знищення матеріальних цінностей [6]. Вибір способу НСД залежить від конкретних цілей та обставин, і може включати в себе різноманітні техніки, такі як хакінг, соціальний інженеринг, фішинг, використання програмних вразливостей тощо (рис. 1.1). Необхідно враховувати, що повний обсяг інформації про конкурента

не може бути отриманий через один єдиний метод, і зазвичай зловмисники використовують комбінацію різних методів для досягнення своїх цілей.

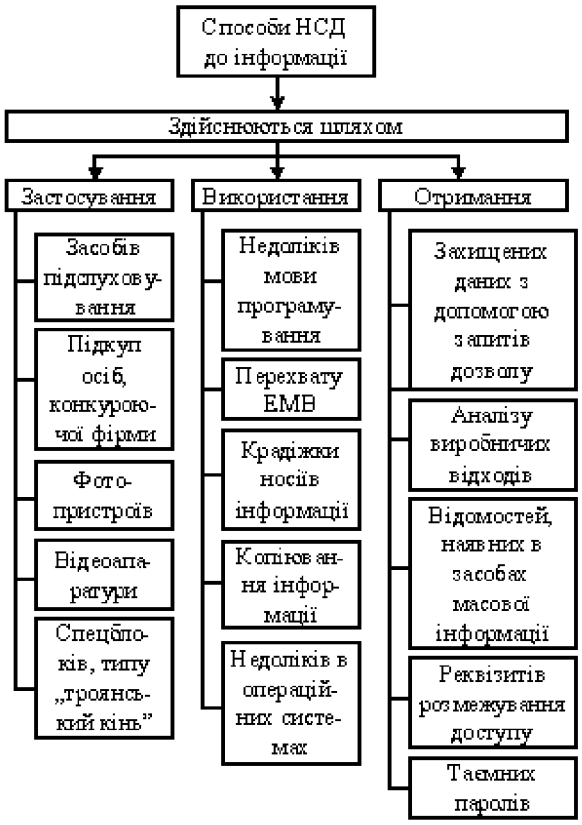

Рисунок 1.1 – Способи НСД до конфіденційної інформації

Принципи забезпечення інформаційної безпеки включають законність, збалансованість інтересів особи, суспільства і держави, комплексність, системність, інтеграцію з міжнародними системами безпеки та економічну ефективність.

Неможливо створити систему, яка була б абсолютно непроникною. Основним принципом є створення механізму захисту, вартість злому якого була б набагато вище, ніж ціна інформації, яку можна отримати. Тому важливо впроваджувати програмні засоби безпеки, що є невід'ємною частиною програмного забезпечення системи та необхідними для здійснення захисних функцій.

Як відзначає експерт з кібербезпеки Дмитро Ганжело, «Усунення наслідків кібератак часто обходиться в кілька разів дорожче, аніж профілактика боротьби

з ними». У сучасних умовах неможливо забезпечити стабільний економічний розвиток без належного захисту інформації, як на рівні окремих підприємств, так і на рівні держави.

1.2 Основні програмно – технічні заходи

Основні програмно – технічні заходи у сфері інформаційної безпеки спрямовані на забезпечення конфіденційності, цілісності та доступності інформації. Деякі з цих заходів включають [7]:

– шифрування даних: захист інформації шляхом перетворення її у незрозумілу форму за допомогою криптографічних алгоритмів;

– автентифікація: визначення і перевірка ідентичності користувачів або систем;

– авторизація: контроль доступу до різних ресурсів або функціональності системи;

– аудит і моніторинг: систематична перевірка подій і дій в системі для виявлення можливих загроз або вразливостей;

– захист від вразливостей: проведення аналізу вразливостей та прийняття заходів для їх запобігання або ліквідації.

Основні поняття програмно – технічного рівня інформаційної безпеки включають:

– кодування: процес перетворення даних у вигляд, який необхідно розуміти або обробляти, для забезпечення конфіденційності;

– шифрування: застосування криптографічних методів для захисту даних від несанкціонованого доступу;

– ідентифікація та автентифікація: визначення і підтвердження ідентичності користувачів або систем;

– контроль доступу: управління правами доступу користувачів до різних ресурсів;

– аудит безпеки: збір і аналіз журналів подій для виявлення атак або порушень безпеки.

До загроз безпеки комп'ютерної мережі відносяться (рис. 1.2):

– загрози розкриття: вони включають несанкціонований доступ до інформації і мережевих ресурсів, перехоплення і ознайомлення з переданими даними по каналах зв'язку, а також фальсифікацію повідомлень і відмову визнавати факт отримання інформації;

– загрози цілісності: загрози включають в себе розкриття і модифікацію даних і програм, копіювання інформації, а також розробку та поширення комп'ютерних вірусів або введення в програмне забезпечення логічних бомб;

– загрози відмови в обслуговуванні: це може включати крадіжку магнітних носіїв і розрахункових документів, руйнування архівної інформації або навмисне її знищення, а також перехоплення та ознайомлення з інформацією, переданою по каналах зв'язку.

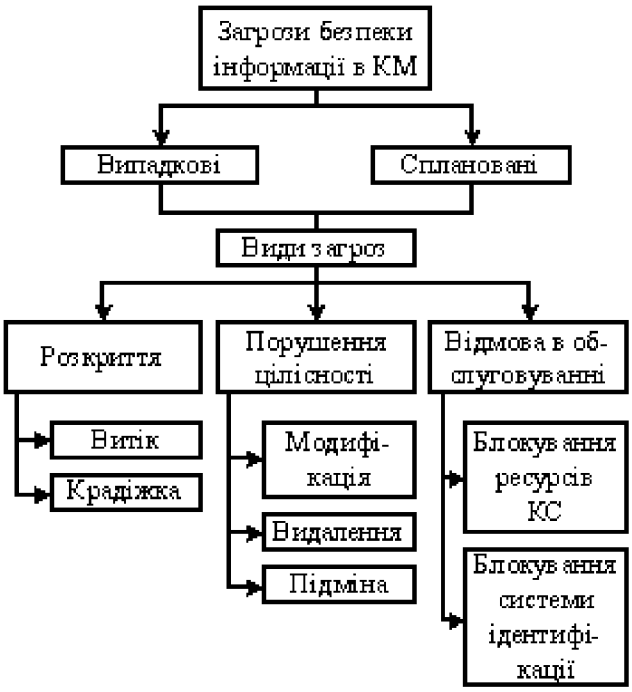

Рисунок 1.2 – Види загроз безпеки інформації в комп'ютерних мережах

Сучасні інформаційні системи мають ряд особливостей, які є важливими з точки зору безпеки [8]:

– розподілені середовища: інформація може зберігатися та оброблюватися в різних географічних місцях або на різних пристроях, що потребує забезпечення безпеки під час передачі та зберігання;

– мобільність: користувачі можуть отримувати доступ до інформації з різних місць і пристроїв, що створює нові виклики щодо безпеки, зокрема втрати пристроїв або зловживання правами доступу;

– широкомасштабність: інформаційні системи можуть обробляти великі обсяги даних, що потребує ефективного управління безпекою та захисту від атак.

Архітектурна безпека включає в себе планування та розробку системи з урахуванням вимог безпеки з самого початку. Це включає в себе:

– визначення загроз: аналіз потенційних загроз і вразливостей, які можуть вплинути на систему;

– розробка заходів захисту: встановлення заходів безпеки, що включають в себе як технічні, так і організаційні заходи;

– реалізація контролю доступу: встановлення механізмів для контролю доступу до ресурсів системи.

Сервіси безпеки зазвичай включають в себе набір інструментів та послуг, спрямованих на забезпечення безпеки інформаційних систем. Деякі з них можуть включати антивіруси, системи виявлення вторгнень, захист від DDoS – атак, керування ідентичністю та управління доступом, сервіси шифрування та інші. Ці сервіси допомагають у виявленні, запобіганні та вирішенні проблем безпеки в інформаційних системах [9].

1.3 Поняття інформаційної безпеки

Інформаційна безпека – це захист інформації від несанкціонованого доступу, використання, руйнування, модифікації або розповсюдження. Це поняття охоплює широкий спектр заходів, технологій, політик та процедур,

спрямованих на забезпечення конфіденційності, цілісності та доступності інформації [10].

Основні складові інформаційної безпеки:

– конфіденційність: захист від несанкціонованого доступу до інформації;

– цілісність: забезпечення того, щоб інформація залишалася непошкодженою та не зміненою;

– доступність: гарантування доступу до інформації для авторизованих користувачів в потрібний час.

В інформаційній ері, де практично всі аспекти життя залежать від комп'ютерних систем і мереж, інформаційна безпека стає критично важливою. Порушення безпеки може призвести до витоку конфіденційної інформації, фінансових втрат, пошкодження репутації та навіть загрози життю.

Проблема інформаційної безпеки складна через постійний розвиток технологій, нових загроз та відповідність правил та законів.

Причини існування комп'ютерних злодіїв [11, 12]:

Комп'ютерні злодії можуть мати різноманітні мотиви, включаючи отримання фінансової вигоди, втручання в політичні процеси, відшкодування особистих образ та навіть тероризм. Мотивації можуть бути економічними, політичними, соціальними або особистими.

Законодавчий, адміністративний і процедурний рівні:

– законодавчий рівень: включає в себе прийняття законів та правил, які регулюють захист інформації та покарання за її порушення;

– адміністративний рівень: охоплює політики та процедури, що встановлюються в організаціях для забезпечення безпеки інформації;

– процедурний рівень: визначає конкретні кроки, які потрібно виконати для виконання політик та правил і забезпечення безпеки інформації.

Додаткові заходи захисту програмного забезпечення можуть включати:

– патчі і оновлення: регулярне встановлення патчів та оновлень для програмного забезпечення, що містить виправлення виявлених вразливостей;

– фірмовий антивірус та антишпигунське програмне забезпечення: використання спеціального програмного забезпечення для виявлення та видалення шкідливого програмного забезпечення, такого як віруси, трояни, шпигунське програмне забезпечення тощо;

– фірмовий файрвол: застосування файрвола для контролю потоку даних між комп'ютерами у мережі та моніторингу вхідних і вихідних з'єднань для виявлення потенційно небезпечних або незаконних дій;

– механізми резервного копіювання: регулярне створення резервних копій даних для відновлення інформації у випадку втрати або пошкодження;

– сегментація мережі: розділення комп'ютерних мереж на підмережі з метою обмеження доступу до ресурсів та зменшення ризику вірусних атак або внутрішніх загроз;

– навчання та освіта користувачів: проведення навчання та інструктажу користувачів щодо правил безпеки в мережі, розпізнавання фішингових атак, небезпечних вебсайтів та інших потенційно небезпечних ситуацій;

– інцидентний менеджмент: розробка процедур для реагування на інциденти безпеки, включаючи виявлення, аналіз та відновлення після інциденту;

Забезпечення інформаційної безпеки є складним завданням, яке вимагає постійного вдосконалення та адаптації до нових загроз і технологій. Тільки завдяки цілісному підходу та усвідомленню всіх сторін суспільства можна забезпечити надійний захист інформації в сучасному цифровому середовищі.

1.4 Стандарти та специфікації в галузі інформаційної безпеки

У галузі інформаційної безпеки існують різноманітні стандарти, специфікації та оціночні моделі, які допомагають організаціям та фахівцям у забезпеченні безпеки інформаційних систем, ці методи, що часто згадуються у публікаціях, спрямовані на захист кібернетичного середовища користувача або

організації [13]. Це середовище включає користувачів, мережі, пристрої, програмне забезпечення, процеси, інформацію в режимі зберігання або передачі, програми, служби та системи, які можуть бути прямо чи опосередковано підключені до мережі. Головна мета полягає в зниженні ризиків, включаючи попередження або пом'якшення кібератак. Ці матеріали включають набори інструментів, політику безпеки, концепції безпеки, гарантії безпеки, керівні принципи, підходи до управління ризиками, навчання, найкращі практики, забезпечення та технології [14].

Стандарти та специфікації в галузі інформаційної безпеки (ІБ) – це документи, що описують кращі практики та рекомендації щодо захисту інформації. Їх дотримання може допомогти організаціям:

– знизити ризики, пов'язані з кібер – атаками, витоками даних, втратою інформації та іншими інцидентами ІБ;

– підвищити довіру клієнтів, партнерів та інших зацікавлених сторін;

– виконати нормативні вимоги, такі як GDPR, CCPA та інші закони про захист даних;

– покращити загальний стан ІБ в організації.

Стандарти та специфікації в галузі інформаційної безпеки включають [15, 16]:

– помаранчева книга (Orange Book): офіційний документ, що визначає стандарти безпеки для комп'ютерних систем, видається Національним Інститутом Стандартів та Технологій США (NIST). Використовується для класифікації та оцінки безпеки операційних систем. Помаранчева книга є одним із найважливіших оціночних стандартів в галузі інформаційної безпеки;

– стандарт ISO/IEC 15408 ("Критерії оцінки безпеки інформаційних технологій"): цей стандарт, відомий також як Common Criteria (Загальні Критерії), надає міжнародні засади для оцінки безпеки продуктів та систем інформаційних технологій. Common Criteria визначає рівні безпеки, які можуть бути використані для оцінки рівня безпеки системи;

– політика безпеки: документ, який визначає загальні правила, процедури та вимоги щодо захисту інформації в організації. Політика безпеки зазвичай включає в себе визначення правил доступу, управління паролями, заходів забезпечення безпеки мережі тощо;

– механізми безпеки: конкретні технічні засоби, які використовуються для захисту інформації, такі як шифрування даних, системи аутентифікації, файрволи тощо;

– рівень гарантованості (Assurance Level): показник, що визначає ступінь довіри до безпеки інформаційної системи або продукту. Рівні гарантованості зазвичай визначаються в рамках оціночних моделей, таких як Common Criteria;

– класи безпеки: використовуються для класифікації систем за рівнем їх захищеності. Наприклад, в Помаранчевій книзі існує ряд класів безпеки від D до A1;

– інформаційна безпека розподілених систем: галузь, що досліджує заходи безпеки, специфічні для розподілених систем, таких як мережі, хмарні обчислення тощо;

– рекомендації X.800: міжнародний стандарт, що визначає загальні принципи мережевої безпеки та основні терміни, які використовуються в цій галузі.

Ці стандарти, специфікації та оціночні моделі є важливими інструментами для розробки, оцінки та забезпечення безпеки інформаційних систем у всьому світі.

1.5 Аналіз безпеки транспортного рівня (TLS) в програмуванні

TLS (Transport Layer Security) – це криптографічний протокол, який використовується для забезпечення конфіденційності та цілісності даних під час їх передачі між двома хостами. Він використовується в широкому спектрі

програм, включаючи веббраузери, електронну пошту, месенджери та багато іншого [17].

Існують два основних типи TLS:

– tls 1.x: старіша версія TLS, яка має кілька відомих вразливостей;

– tls 1.3: нова версія TLS, яка більш безпечна і ефективна.

Переваги використання TLS:

– дані шифруються під час передачі, тому їх не можуть прочитати сторонні особи;

– дані не можуть бути змінені під час передачі без відома одержувача;

– сервери та клієнти можуть аутентифікувати один одного, щоб запобігти спуфінгової атаки.

Вразливості TLS:

– якщо відкритий ключ сервера скомпрометований, зловмисник може перехопити та розшифрувати дані;

– деякі шифри, що використовуються в TLS, можуть бути зламані зловмисниками;

– неправильна конфігурація TLS може призвести до вразливостей.

TLS може використовувати розширення для додаткових функцій, таких як [18]:

– server name indication (SNI): дозволяє клієнту вказувати ім'я сервера, до якого він хоче підключитися, під час початкового рукостискання;

– application layer protocol negotiation (ALPN): дозволяє клієнту та серверу узгоджувати протокол прикладного рівня, який буде використовуватися після встановлення з'єднання TLS;

– elliptic curve cryptography (ECC): забезпечує більш стійке до атак шифрування, ніж традиційні алгоритми RSA;

– session tickets: дозволяє клієнтам відновлювати сеанси TLS без повторного повного рукостискання.

TLS використовує різні алгоритми шифрування та хешування для забезпечення конфіденційності та цілісності даних. Деякими з найпоширеніших алгоритмів є:

– алгоритми шифрування: AES, ChaCha20, Camellia;

– алгоритми хешування: SHA-256, SHA-384, MD5.

TLS використовує цифрові сертифікати для автентифікації серверів та клієнтів. Сертифікати TLS містять інформацію про власника сертифіката, його відкритий ключ та інші дані.

Набір шифрів TLS визначає комбінацію алгоритмів шифрування, хешування та аутентифікації, які використовуються для захисту сеансу TLS. Поширені набори шифрів включають:

– ECDHE-RSA-AES256-GCM-SHA384: набір шифрів використовує ECC для аутентифікації, RSA для обміну ключами, AES-256 для шифрування та GCM (Galois/Counter Mode) для аутентифікації повідомлень;

– ECDHE-ECDSA-AES128-GCM-SHA256: цей набір шифрів використовує ECC для аутентифікації, ECDSA (Elliptic Curve Digital Signature Algorithm) для обміну ключами, AES-128 для шифрування та GCM для аутентифікації повідомлень.

TLS 1.3 це найновіша версія протоколу TLS, яка пропонує ряд поліпшень, включаючи:

– покращення продуктивності: TLS 1.3 використовує нові алгоритми та протоколи, які можуть значно покращити продуктивність порівняно з TLS 1.2;

– підвищення безпеки: TLS 1.3 включає нові функції безпеки, такі як PFS (Perfect Forward Secrecy), які роблять атаки на минулі сеанси більш складними;

– спрощення протоколу: TLS 1.3 видаляє деякі застарілі функції та спрощує протокол, що робить його більш стійким до атак.

Існує багато різних типів атак на TLS, таких як:

– атаки на рукостискання: ці атаки націлені на процес рукостискання TLS, щоб перехопити ключі або скомпрометувати автентифікацію;

– атаки на шифрування: націлені на алгоритми шифрування, що використовуються TLS, щоб зламати шифрування та прочитати дані;

– атаки на сертифікати: націлені на цифрові сертифікати, що використовуються TLS, щоб скомпрометувати автентифікацію або підробити сертифікати.

TLS – це важливий протокол, який використовується для забезпечення безпеки даних під час їх передачі. Важливо використовувати TLS правильно, щоб захистити ваші дані від крадіжки, модифікації та перехоплення.

РОЗДІЛ 2
АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СТВОРЕННЯ СЕРТИФІКАТУ X.509, З РОЗШИРЕННЯМИ АЛЬТЕРНАТИВНОГО ІМЕНІ СУБЪЕКТА (SAN)

2.1 Мета та задачі програмного забезпечення для створення сертифікатів X.509 з розширеннями SAN

Програмне забезпечення для створення сертифікатів X.509 з розширеннями SAN (Subject Alternative Name) використовується для генерування цифрових сертифікатів, які містять кілька альтернативних імен хостів або доменних імен. Ці сертифікати можуть використовуватися для захисту декількох вебсайтів або серверів за допомогою одного сертифіката.

Основна мета ПЗ для створення сертифікатів X.509 з розширеннями SAN [19]:

– зниження витрат: один сертифікат SAN може використовуватися для захисту декількох вебсайтів або серверів, що може заощадити гроші на купівлі окремих сертифікатів для кожного з них;

– зручність керування: легше керувати одним сертифікатом SAN, ніж кількома окремими сертифікатами;

– зменшення ризику помилок: використання одного сертифіката SAN зменшує ризик помилок при встановленні та налаштуванні сертифікатів.

Програмне забезпечення для створення сертифікатів X.509 з розширеннями SAN зазвичай виконує такі задачі:

– генерація пар ключів: програмне забезпечення генерує пару ключів (відкритий ключ і закритий ключ). Відкритий ключ включається до сертифіката, а закритий ключ зберігається в секреті;

– створення запиту на сертифікат: програмне забезпечення створює запит на сертифікат, який містить інформацію про власника сертифіката, а також список альтернативних імен хостів або доменних імен;

– надсилання запиту на сертифікат до ЦС: запит на сертифікат надсилається до центру сертифікації (ЦС);

– отримання сертифіката від ЦС: ЦС перевіряє запит на сертифікат і, якщо все в порядку, видає сертифікат;

– встановлення сертифіката: сертифікат встановлюється на вебсервер або інший пристрій, який його використовуватиме.

Існує багато програмних продуктів для створення сертифікатів X.509 з розширеннями SAN. До популярних варіантів належать [20]:

– OpenSSL: безкоштовний пакет з відкритим кодом, який включає інструменти для створення сертифікатів X.509;

– Venafi: комерційне програмне забезпечення для керування сертифікатами, яке включає функції для створення сертифікатів X.509 з розширеннями SAN;

– Entrust Datacard PKI Platform: ще одне комерційне програмне забезпечення для керування сертифікатами, яке включає функції для створення сертифікатів X.509 з розширеннями SAN.

При виборі програмного забезпечення для створення сертифікатів X.509 з розширеннями SAN слід врахувати такі фактори:

– вартість: деякі програмні продукти є безкоштовними, а інші платними;

– функції: програмні продукти пропонують більше функцій, ніж інші;

– простота використання: деякі програмні продукти простіші у використанні, ніж інші;

– підтримка: програмні продукти пропонують кращу підтримку, ніж інші.

Програмне забезпечення для створення сертифікатів X.509 з розширеннями SAN може бути цінним інструментом для організацій, яким потрібно захистити декілька вебсайтів або серверів за допомогою одного сертифіката. При виборі програмного забезпечення важливо врахувати свої потреби та бюджет.

2.2 Архітектура програмного забезпечення для створення сертифікатів

Програмне забезпечення для створення сертифікатів X.509 з розширеннями SAN зазвичай має модульну архітектуру, яка складається з наступних компонентів [21]:

– модуль генерації ключів: відповідає за генерування пар ключів (відкритий ключ і закритий ключ);

– модуль створення запиту на сертифікат: модуль відповідає за створення запиту на сертифікат, який містить інформацію про власника сертифіката, а також список альтернативних імен хостів або доменних імен;

– модуль зв'язку з ЦС: відповідає за зв'язок з центром сертифікації (ЦС) для надсилання запиту на сертифікат та отримання сертифіката;

– модуль зберігання: відповідає за зберігання закритого ключа та сертифіката;

– інтерфейс користувача: модуль забезпечує інтерфейс для користувачів для взаємодії з програмним забезпеченням.

Детальний опис компонентів:

– модуль генерації ключів: використовує криптографічні алгоритми для генерування пари ключів (відкритий ключ і закритий ключ). Відкритий ключ використовується для шифрування даних, а закритий ключ використовується для дешифрування даних. Важливо захищати закритий ключ, оскільки його можна використовувати для створення підроблених сертифікатів;

– модуль створення запиту на сертифікат: збирає інформацію про власника сертифіката, а також список альтернативних імен хостів або доменних імен. Ця інформація використовується для створення запиту на сертифікат у форматі PKCS10;

– модуль зв'язку з ЦС: використовує протокол SSL/TLS для встановлення безпечного з'єднання з ЦС. Потім модуль надсилає запит на сертифікат ЦС і отримує сертифікат у відповідь;

– модуль зберігання: зберігає закритий ключ і сертифікат в безпечному місці. Закритий ключ зазвичай зберігається у файлі на диску, а сертифікат зазвичай зберігається у сховищі сертифікатів операційної системи;

– інтерфейс користувача: надає користувачам інтерфейс для введення інформації про власника сертифіката, вибору альтернативних імен хостів або доменних імен, а також для створення, імпорту та експорту сертифікатів.

Модульна архітектура має ряд переваг, зокрема:

– гнучкість: модулі можна легко додавати, видаляти або замінювати, що дозволяє адаптувати програмне забезпечення до мінливих потреб;

– простота обслуговування: можна легко оновлювати або виправляти, не впливаючи на інші частини програмного забезпечення;

– повторне використання: модулі можна повторно використовувати в інших програмах.

Модульна архітектура є поширеним вибором для програмного забезпечення. Ця архітектура забезпечує гнучкість, простоту обслуговування та можливість повторного використання, що робить її ідеальною для складних програмних продуктів.

2.3 Функціональність компонентів

Генерація криптографічно стійких пар ключів (RSA, EC) [22]:

– використовує алгоритми RSA та EC, рекомендовані NIST, для генерування пар ключів відповідної довжини, що гарантує стійкість до сучасних методів злому;

– забезпечує гнучкість у виборі алгоритму та довжини ключа, дозволяючи користувачам підбирати оптимальні параметри з урахуванням потреб та рівня безпеки.

Безпечне зберігання приватних ключів [23]:

– застосовує надійні методи захисту, такі як апаратні модулі безпеки (HSM) або шифрування на диску, для забезпечення конфіденційності та цілісності приватних ключів;

– використовує криптографічні алгоритми та протоколи, стійкі до атак типу брутфорс та інших методів розкриття ключа.

Надання відкритих ключів:

– надає відкриті ключі іншим компонентам програмного забезпечення та стороннім учасникам у зручному форматі, наприклад PEM або DER;

– забезпечує цілісність та автентичність відкритих ключів, використовуючи криптографічні підписи або інші методи верифікації.

Формування CSR:

– створює структуру CSR (Certificate Signing Request) відповідно до стандарту X.509, що включає всі необхідні атрибути суб'єкта, такі як ім'я, організація, країна тощо;

– кодує атрибути суб'єкта у форматі ASN.1, який використовується для стандартизованого представлення даних у сертифікатах X.509;

– забезпечує правильне форматування та структуру CSR, що гарантує його прийняття Центром сертифікації (CA).

Додавання розширень SAN:

– додає до CSR розширення SAN (Subject Alternative Name), які дозволяють пов'язати з одним сертифікатом X.509 кілька доменних імен, IP-адрес або електронних адрес;

– підтримує різні типи SAN, включаючи DNS – імена, IP-адреси IPv4 та IPv6, а також адреси електронної пошти;

– забезпечує коректне кодування та декодування розширень SAN у форматі ASN.1, що гарантує їх правильне тлумачення CA.

Підписання CSR:

– підписує CSR приватним ключем суб'єкта, використовуючи криптографічний алгоритм підпису, наприклад SHA-256 з RSA;

– забезпечує цілісність та автентичність CSR, що гарантує, що запит на сертифікат не був змінений або підроблений;

– дозволяє CA перевірити автентичність суб'єкта та підтвердити його право на отримання сертифіката X.509.

Встановлення з'єднання з CA:

– використовує безпечні протоколи HTTPS або LDAP для встановлення з'єднання з Центром сертифікації (CA);

– забезпечує автентифікацію та шифрування даних під час з'єднання, що гарантує конфіденційність та захист від перехоплення;

– використовує сертифікати CA для верифікації автентичності сервера та захисту від атак типу "людина посередині".

Надсилання CSR до CA:

– відправляє підписаний CSR до CA, використовуючи безпечний протокол передачі даних;

– забезпечує цілісність та автентичність CSR під час передачі, що гарантує, що запит не був змінений або підроблений;

– використовує методи стиснення даних, якщо це можливо, для оптимізації пропускної здатності каналу зв'язку.

Отримання та обробка сертифіката X.509 від CA [24]:

– отримує від CA сертифікат X.509, виданий у відповідь на CSR;

– перевіряє автентичність сертифіката CA, використовуючи його відкритий ключ та довірений кореневий сертифікат;

– валідує термін дії сертифіката та перевіряє його відсутність в списках відкликаних сертифікатів (CRL);

– записує отриманий сертифікат X.509 та ланцюжок сертифікатів у надійне сховище.

Зберігання сертифікатів, ланцюжків сертифікатів та CRL:

– зберігає сертифікати X.509, ланцюжки сертифікатів (послідовність сертифікатів до кореневого) та списки відкликаних сертифікатів (CRL) у безпечному сховищі;

– підтримує різні формати сертифікатів, такі як PEM та DER, забезпечуючи легкий доступ та обмін;

– використовує методи контролю доступу для обмеження можливості несанкціонованого використання або зміни сертифікатів.

Надання доступу до сертифікатів:

– надає іншим компонентам програмного забезпечення та авторизованим користувачам доступ до сертифікатів, ланцюжків сертифікатів та CRL;

– забезпечує різні методи доступу, такі як API або графічний інтерфейс, залежно від потреб користувачів;

– може надавати обмежений доступ до певних атрибутів сертифіката, якщо необхідно.

Керування сертифікатами:

– дозволяє оновлювати сертифікати до їх закінчення дії, автоматизуючи процес перевипуску;

– надає функції видалення та відкликання сертифікатів у разі їх компрометації або зміни власника;

– може відстежувати термін дії сертифікатів та генерувати попередження про необхідність оновлення.

2.4 Реалізація програмного забезпечення

Вибір мови програмування та інструментів для реалізації програмного забезпечення для створення сертифікатів залежить від декількох факторів, таких як:

– функціональні вимоги: обсяг та складність програмного забезпечення, а також необхідні функції, такі як підтримка різних алгоритмів криптографії, форматів сертифікатів та протоколів зв'язку;

– платформа: операційна система, на якій буде працювати програмне забезпечення, а також апаратні ресурси, доступні для його виконання;

– досвід розробників: знання та навички команди розробників у роботі з певними мовами програмування та інструментами;

– доступність бібліотек та фреймворків: наявність бібліотек та фреймворків, які можуть спростити розробку та тестування програмного забезпечення.

Ось кілька популярних мов програмування та інструментів, які можна використовувати:

– Python: універсальна мова програмування з великою кількістю бібліотек та фреймворків для криптографії та роботи з сертифікатами, таких як cryptography та OpenSSL;

– Java: надійна та масштабована мова програмування, яка добре підходить для розробки складних програмних систем. В ній існують бібліотеки, такі як Bouncy Castle та JSSE, які надають функціональність для роботи з сертифікатами X.509;

– C/C++: мови програмування, які пропонують високу продуктивність та контроль над пам'яттю, що може бути важливо для криптографічних операцій. Бібліотеки, такі як OpenSSL та Cryptography, доступні для роботи з сертифікатами X.509;

2.5 Створення сертифікату x.509, з розширеннями альтернативного імені субъекта

Налаштовую файл server.openssl.cnf для подальшого створення CSR з декількома іменами (рис. 2.1).

Open

server.openssl.cnf
~/Завантаження/Labsetup/volumes/client-certs

```
1 [ req ]
2 prompt  =  no
3 distinguished_name = req_distinguished_name
4 req_extensions  =  req_ext
5
6 [ req_distinguished_name ]
7 C = UA
8 ST = Poltavska Obl.
9 L = Poltava
10 O = Mulko
11 CN = www.mulko2024.com
12
13 [ req_ext ]
14 subjectAltName = @alt_names
15
16 [alt_names]
17 DNS.1  =  www.mulko2024.com
18 DNS.2  =  www.mulko2024A.com
19 DNS.3  =  www.mulko2024B.com
```

Рисунок 2.1 – Налаштування файлу для створення CSR

Використовую кореневий файл CA ca.crt для підпису сертифіката з декількома іменами хостів. Після цього я вивожу файл server_alt.crt і файл server_alt.key, потім використовую ці два файли [25]. Файл було додано до шляху сертифікації в програмі server.py для тестування (рис. 2.2).

```
andrew@Andrew:~/Завантаження/Labsetup (копія)$ opensl req -newkey rsa:2048 -co
nfig ./server.openssl.cnf -batch -sha256 -keyout server.key -out server.csr
.+.....+....+.........+.....+.+.........+..........+...+..+.......+...+..........+....
...+..................+..+.............+..+...++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++*.....+.+.....+....+.....+.+.....+......+...+....+......+....
..........+...+....+...+.....................+..+...+.......+...++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++*.+..+............+.+......+...+..+.+.........
........+.............+..............+.+.....+...+......++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++
........+.+..+.........................+.....+....+.....+..........+..+....+...+.....+...
...+.+...+...+..+...+...+.+...+...+..++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++*..................+...+....+..++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++*......+.....+....+...+....+.....+......+.........+.+..+............
..+...+............+.....+.+..............+......+.........+......+.+..............+.
+.........+......+....+......+.....+...+.............+..+......+.+.........+...........
...+......+.+.....+.+..+...+....+...+...+.....+.........+...+.......+...+..+....+..+..
..+...+...........+.........+.+.........................+.......+..+............+....+..+..
.+....+..+....+.........+.....+......+.......+..+.......+...++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

Рисунок 2.2 – Створення CSR

Оскільки було імпортовано приватний сертифікат ca.crt у браузер, а згенерований сертифікат сервера виданий цим ca.crt, він пройде перевірку. Як видно, альтернативні імена працюють (рис. 2.3).

Рисунок 2.3 – Результат відкриття серверу з альтернативним ім'ям

Загалом, додавання альтернативних імен до сертифікатів X.509 може бути корисним способом розширення їхньої функціональності та гнучкості.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ TLS

3.1 Вибір засобів розробки для аналізу TLS

Переді мною постало важливе питання – вибір найкращих засобів для всебічного аналізу протоколу TLS в середовищі Linux Ubuntu. Тут існує чимало інструментів, і кожен з них має свої сильні та слабкі сторони.

Одним з найпотужніших рішень є Wireshark – легендарний аналізатор мережевого трафіку з багатими можливостями. Його графічний інтерфейс дозволяє захоплювати і розбирати пакети на різних рівнях моделі OSI, включаючи транспортний та прикладний. Wireshark здатний розшифровувати TLS – трафік за наявності відповідних сертифікатів, даючи змогу вивчати зміст захищених пакетів. Фільтрація, кольорове кодування та зручна ієрархія протоколів полегшують навігацію та аналіз. Хоча потужний, Wireshark досить простий у використанні та має багату базу знань користувачів [26].

Проте, часом зручніше працювати з консольними утилітами, такими як легендарний tcpdump. Цей невибагливий інструмент дозволяє захоплювати пакети з мережевих інтерфейсів та зберігати їх у файлах для подальшого аналізу іншими засобами. Хоча сам tcpdump не може дешифрувати TLS, його виходи можна аналізувати за допомогою OpenSSL чи того ж Wireshark. Кросплатформність та мінімалізм є перевагами tcpdump для ситуацій, коли не потрібен повноцінний графічний інтерфейс.

Кажучи про OpenSSL, не можна не згадати цю могутню криптографічну бібліотеку з відкритим кодом. Її утиліти, такі як s_client та x509, дозволяють тестувати TLS з'єднання, переглядати сертифікати та аналізувати низку параметрів безпеки. OpenSSL є основою багатьох застосунків, працюючих з захищеними з'єднаннями.

Не можна забувати і про спеціалізовані інструменти для аналізу саме TLS протоколу. Серед них варто згадати tls_prober та testssl.sh - ці програми детально

перевіряють налаштування TLS серверів, виявляють вразливості та недоліки в конфігураціях. Зокрема, testssl.sh вражає глибиною аналізу, включаючи підтримувані версії протоколів, шифри, крипто – бібліотеки тощо.

Burp Suite ця потужна платформа включає проксі – сервер для перехоплення HTTPS трафіку, інструменти для скануваннях вразливостей та багато іншого. Burp Suite дозволяє інспектувати, модифікувати та повторювати запити, автоматизуючи процес пошуку дірок у системі безпеки [27].

Розмаїття засобів дозволяє обрати оптимальний інструментарій відповідно до потреб проєкту та рівня експертизи команди. Утиліти з консолі, як tcpdump та OpenSSL, ідеальні для швидких перевірок та автоматизованих сценаріїв. Для більш глибокого аналізу я б рекомендував Wireshark як потужну настільну програму з величезною функціональністю. А спеціалізовані tls_prober і testssl.sh стали б у нагоді для аудиту налаштувань та пошуку вразливостей у TLS конфігураціях. Burp Suite візьме на озброєння команда тестувальників, зосереджена на комплексному аналізі безпеки вебдодатків.

3.2 TLS рукостискання

Встановив TCP зв'язок з https://example.com за допомогою handshake.py (рис. 3.1). Був використаний лістинг для програми.

```
#!/usr/bin/env python3
import socket, ssl, sys, pprint
hostname = sys.argv[1]
port = 443
cadir = '/etc/ssl/certs'
#     Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(capath=cadir) context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True
#     Create TCP connection
```

```
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
        input("After making TCP connection. Press any key to continue ...")
        #      Add the TLS
        ssock = context.wrap_socket(sock, server_hostname=hostname,
        do_handshake_on_connect=False) ssock.do_handshake() # Start the handshake
pprint.pprint(ssock.getpeercert())
        input("After handshake. Press any key to continue ...")
        # Close the TLS Connection ssock.shutdown(socket.SHUT_RDWR) ssock.close()
```

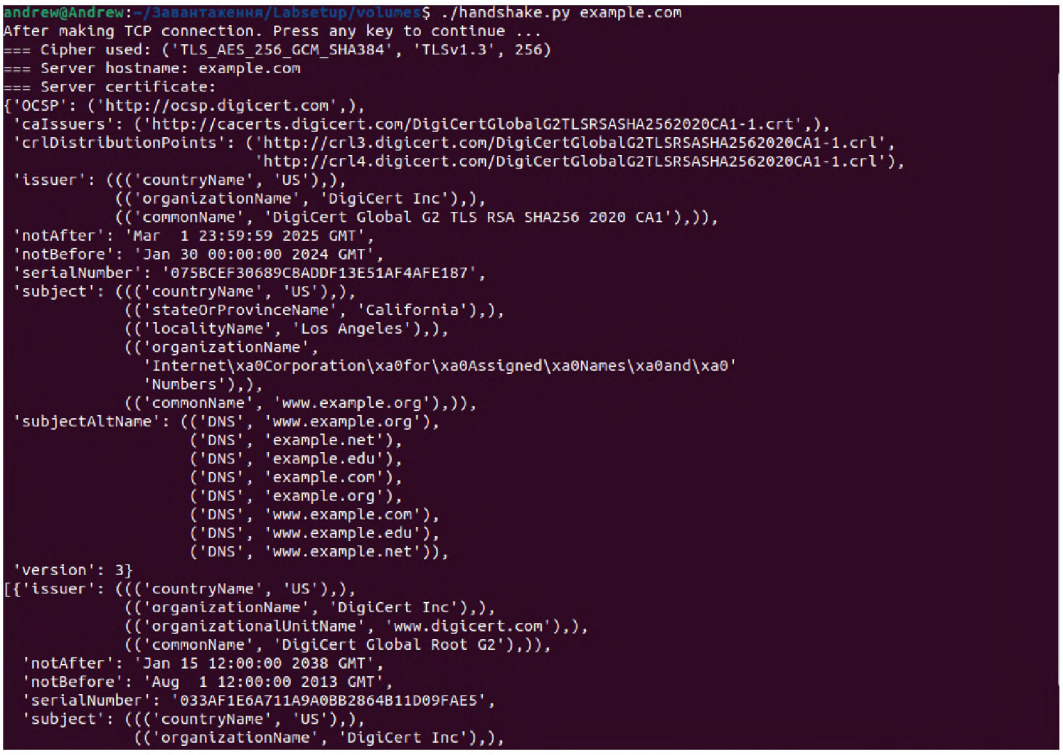

Рисунок 3.1 – Командний рядок після виконання команди handshake.py

TLS вимагає чотирьох кроків для забезпечення безпечного зв'язку для сокетів:

Першим кроком є створення об'єкта контексту TLS. Об'єкт зберігає наші налаштування параметрів для автентифікації сертифіката та вибору алгоритму шифрування [28].

Другим кроком є встановлення протоколу рукостискання TCP, встановлення TCP – з'єднання та частина коду sock.

Третій крок полягає у виклику контекстного об'єкта, створеного на першому кроці, та використанні для нього методу wrap_socket(), що означає, що бібліотека OpenSLL відповідає за контроль нашого TCP – з'єднання. Потім обмін необхідною інформацією рукостискання зі стороною, яка спілкується, і встановлення зашифрованого зв'язку. Частина коду ssock =context.wrap_socket...

Останнім кроком є використання об'єкта ssl_sock, повернутого викликом wrap_socket() для всіх наступних зв'язків. Подальше спілкування має використовувати формат, подібний до імені ssock.method.

TLS_AES_256_GCM_SHA384, TLSv1.3, 256: значення кожного параметра.

Директорія /etc/ssl/certs призначена для зберігання сертифікатів корневих центрів сертифікації (root certificate authorities, root CAs) в операційних системах сімейства Unix/Linux. Ці сертифікати використовуються для перевірки довіри до SSL/TLS сертифікатів вебсайтів, серверів та інших служб, що використовують захищені з'єднання.

Основні функції цієї директорії:

– збереження кореневих сертифікатів: всі корневі сертифікати, яким довіряє операційна система, зберігаються тут. Це дозволяє встановлювати захищені з'єднання з будь – яким вебсайтом чи службою, сертифікат якої був підписаний одним з цих кореневих центрів сертифікації.

– перевірка довіри: коли програма намагається встановити захищене з'єднання (HTTPS, SSL/TLS), вона перевіряє ланцюжок сертифікатів проти сертифікатів у цій директорії. Якщо кореневий сертифікат знайдено, з'єднання вважається довіреним.

– оновлення сертифікатів: іноді центри сертифікації відкликають або оновлюють свої корневі сертифікати. В такому випадку адміністратори можуть додавати нові або вилучати старі сертифікати з цієї директорії для забезпечення актуальності.

Використовую Wireshark для захоплення мережевого трафіку під час виконання програми (рис. 3.2).

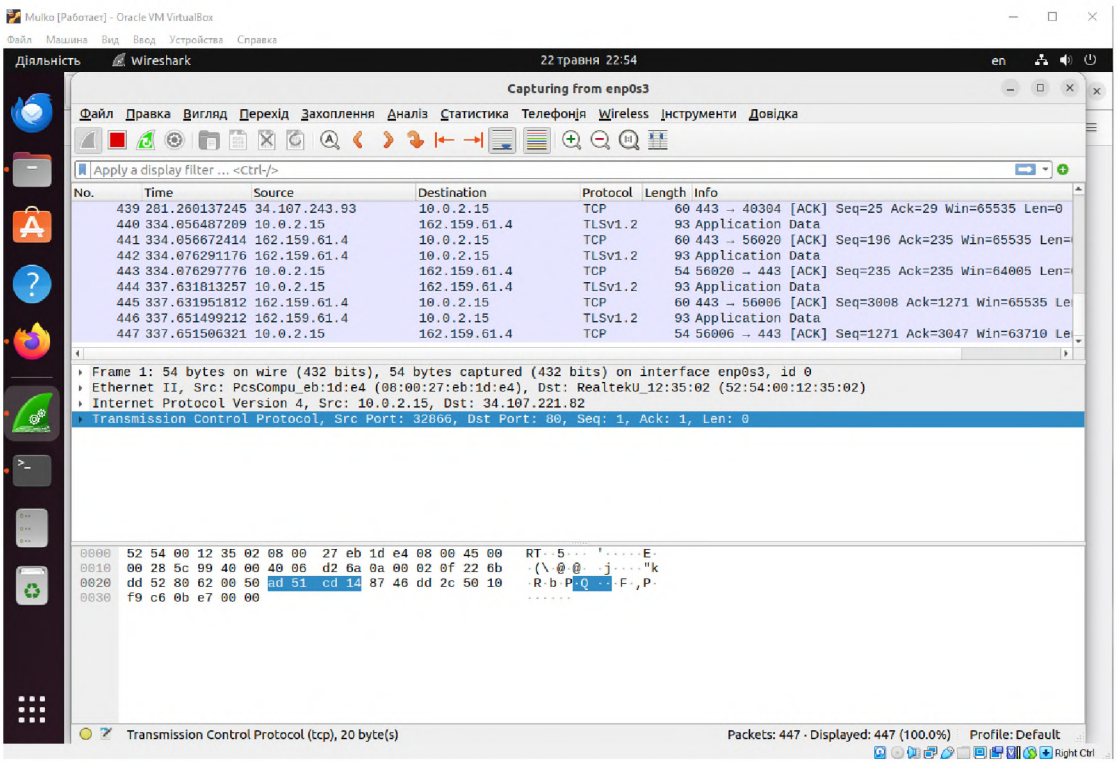

Рисунок 3.2 – Використання середовища Wireshark

Спочатку бачимо кілька пакетів, що містять SYN, SYN – ACK і ACK – це так зване "рукостискання TCP". Воно ініціюється клієнтом, який відправляє SYN (пакет синхронізації) на сервер. Сервер відповідає SYN – ACK, а клієнт підтверджує цю відповідь, відправляючи пакет ACK. Після цього початкового рукостискання TCP встановлюється повне TCP з'єднання.

Після встановлення TCP з'єднання бачимо кілька пакетів, позначених як "TLSv1.2". Це – рукостискання TLS, під час якого клієнт і сервер обмінюються повідомленнями для узгодження параметрів безпечного з'єднання та перевірки автентичності один одного [29].

Рукостискання TLS ініціюється клієнтом, який відправляє "Client Hello". Сервер відповідає з "Server Hello" і своїм сертифікатом. Потім клієнт перевіряє сертифікат сервера і, якщо він дійсний, обмінюється ключами для подальшого шифрування трафіку.

Зв'язок між рукостисканнями TCP і TLS полягає в тому, що TLS працює поверх протоколу TCP. Спочатку потрібно встановити TCP з'єднання, а потім поверх нього ініціюється безпечне TLS з'єднання за допомогою рукостискання

TLS. Завдяки цьому процесу досягається конфіденційність та цілісність даних, що передаються через небезпечну мережу.

Отже, аналізуючи пакети в Wireshark, я спостерігав початкове TCP рукостискання для встановлення з'єднання, а потім рукостискання TLS поверх TCP для налаштування безпечного шифрованого каналу між клієнтом і сервером. TCP-handshake з бажаним сервером (439, 440, 441, 442).

3.3 Створення CA's сертифікату

Змінив шлях до сертифікатів на новостворений каталог client – certs (рис 3.3).

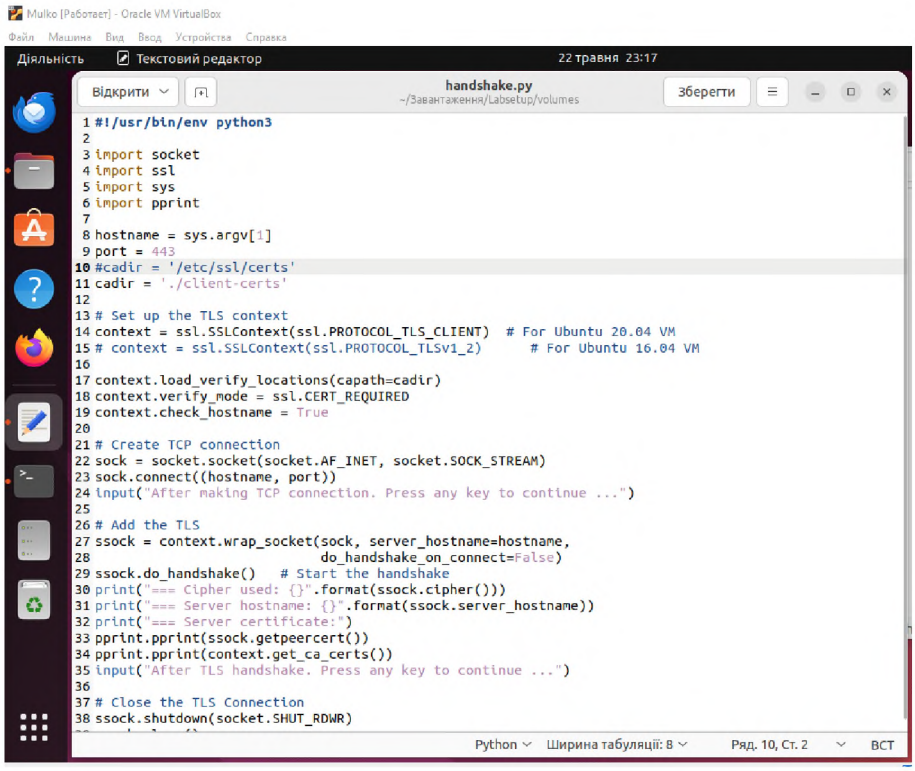

Рисунок 3.3 – Змінення налаштування файлу на новостворений каталог

Після зміни шляху намагаюсь запустити скрипт. Отримую помилку – програма не може знайти локальний сертифікат (рис. 3.4).

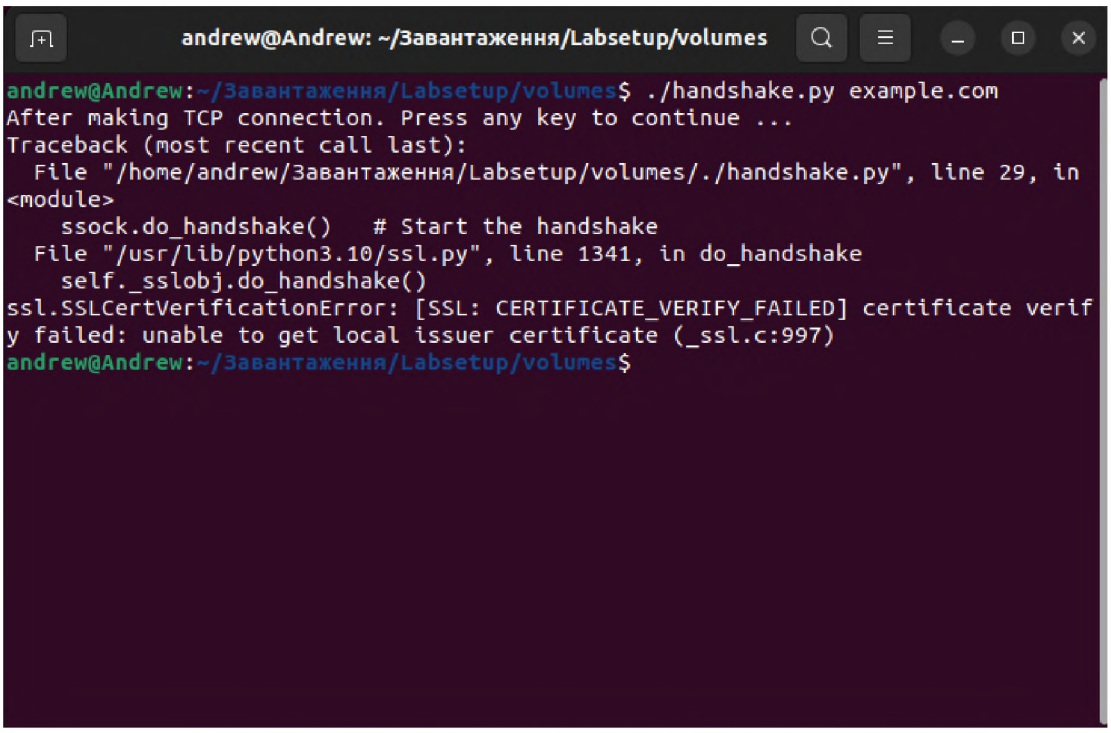

Рисунок 3.4 – Командний рядок після зміни шляху до сертифікатів

Для вирішення проблеми копіюю необхідний сертифікат (DigiCert Global Root CA) в папку client – certs (рис. 3.5).

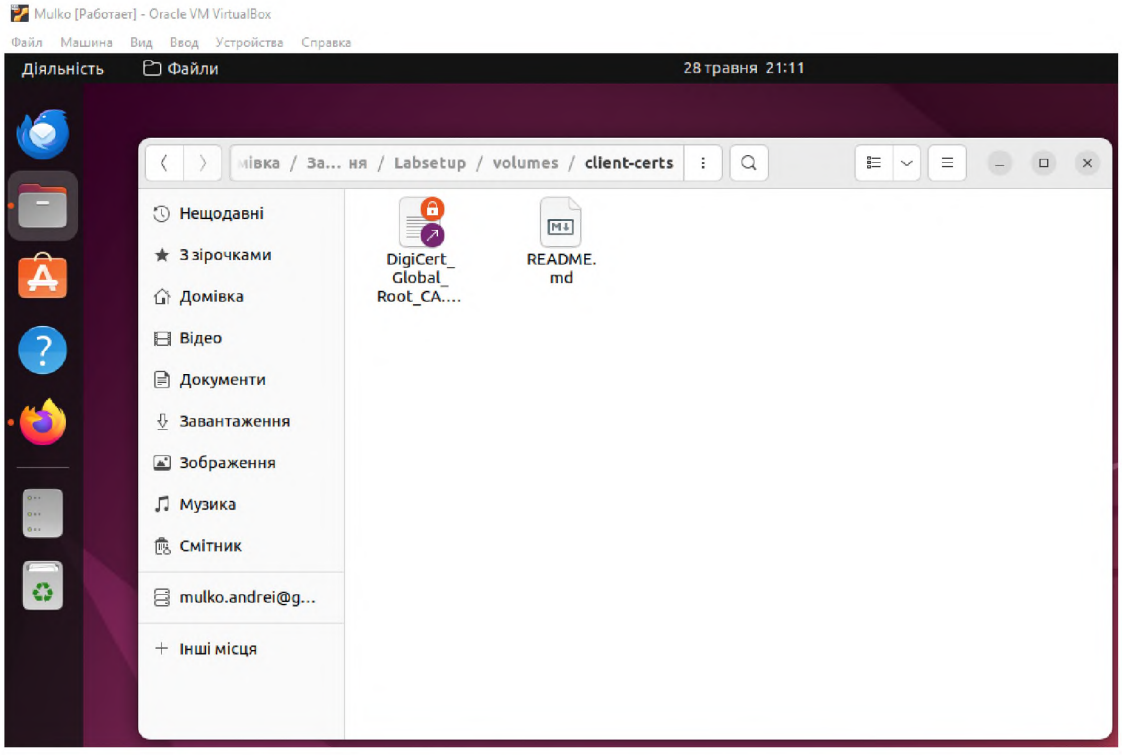

Рисунок 3.5 – Вигляг папки client – certs після додання сертифікату

Генерую хеш – значення, яке потім використовується для створення символьного посилання (рис. 3.6).

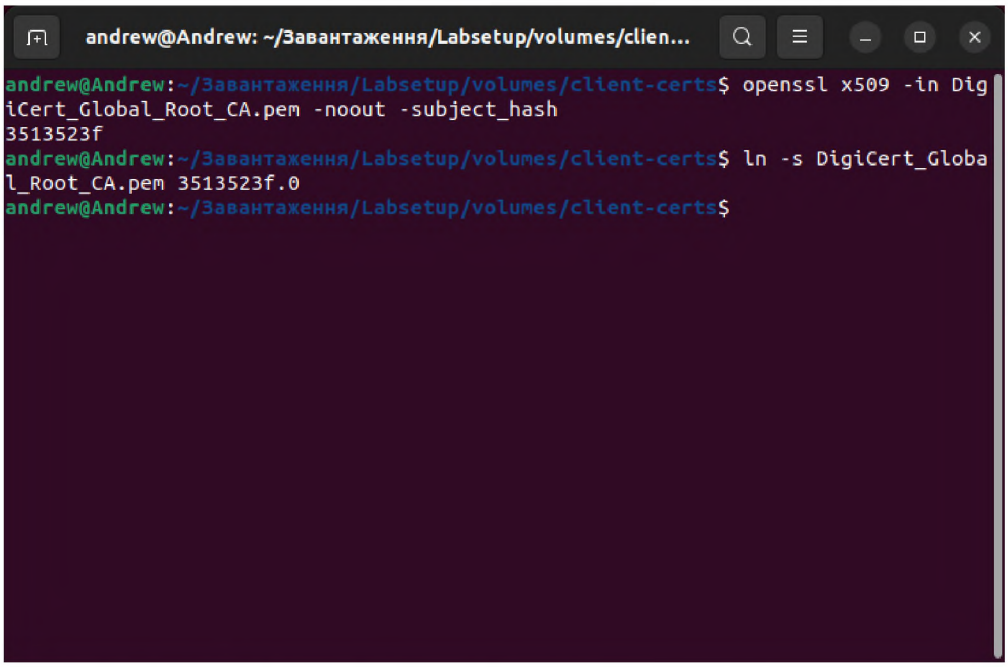

Рисунок 3.6 – Командний рядок при генеруванні хеш – значення

В результаті все запрацювали як повинно. Помилка зникла (рис. 3.7).

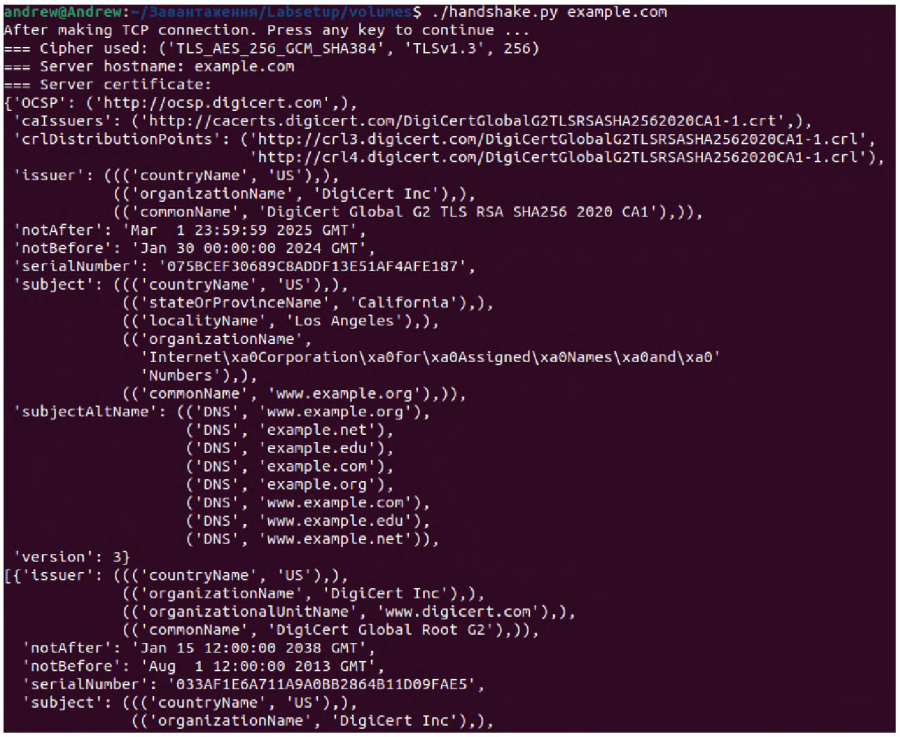

Рисунок 3.7 – Командний рядок після додання сертифікату

За допомогою команди dig дізнаюсь IP-адресу example.com (рис. 3.8).

```
andrew@Andrew:~$ dig example.com

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58838
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            1800    IN      A       93.184.215.14

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue May 28 21:21:07 EEST 2024
;; MSG SIZE  rcvd: 56
```

Рисунок 3.8 – Командний рядок після виконання команти dig

Змінюю файл etc/hosts відповідно до отриманого значення шляхом виконання команди dig (рис. 3.9).

```
Відкрити                    *hosts [Лише читання]
                                /etc
1 127.0.0.1       localhost
2 127.0.1.1       Andrew.Mulko    Andrew
3 93.184.215.14   www.example2024.com
4 # The following lines are desirable for IPv6 capable hosts
5 ::1     ip6-localhost ip6-loopback
6 fe00::0 ip6-localnet
7 ff00::0 ip6-mcastprefix
8 ff02::1 ip6-allnodes
9 ff02::2 ip6-allrouters
```

Рисунок 3.9 – Змінення налаштування файлу до отриманого значення

Результат зі значенням змінної check_hostname = False (рис. 3.10) та True (рис. 3.11).

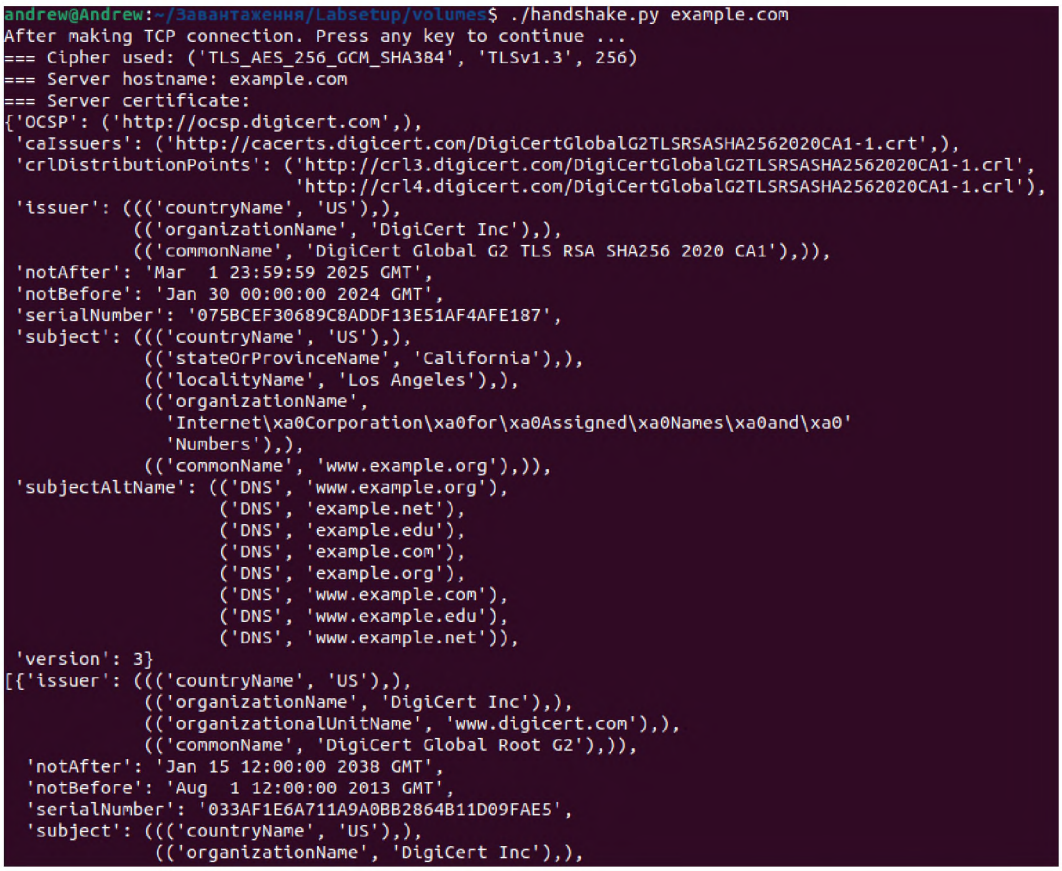

Рисунок 3.10 – Командний рядок зі значенням змінної False

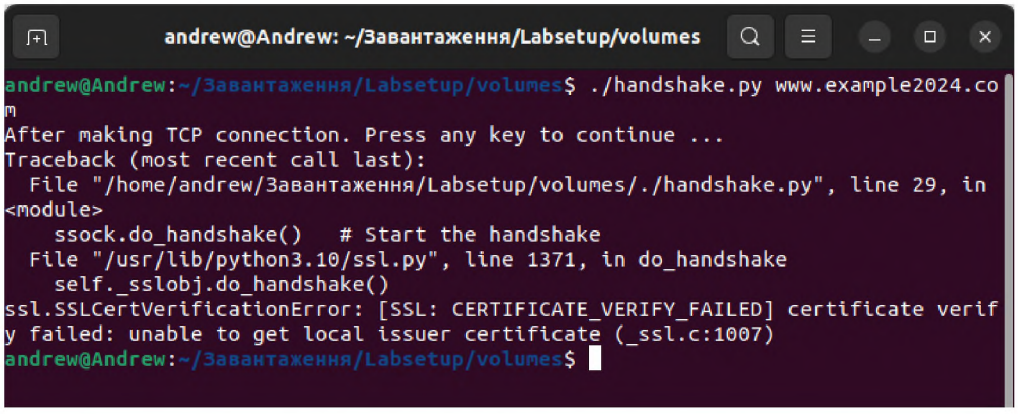

Рисунок 3.11 – Командний рядок зі значенням змінної True

Параметр context.check_hostname є частиною конфігурації SSL/TLS. Якщо встановлено значення True, ім'я хоста в сертифікаті SSL сервера збігається з цільовим призначенням. Якщо встановлено значення False, ця перевірка вимикається. Перевірка імені хоста має вирішальне значення для перевірки автентичності сервера в з'єднаннях SSL/TLS. Вимкнення цієї перевірки ставить

під загрозу безпеку зв'язку, роблячи його чутливим до перехоплення та несанкціонованого доступу [30].

3.4 Впровадження простого серверу TLS

Створюю необхідні сертифікати та розміщаю їх у папці client – certs (рис. 3.12).

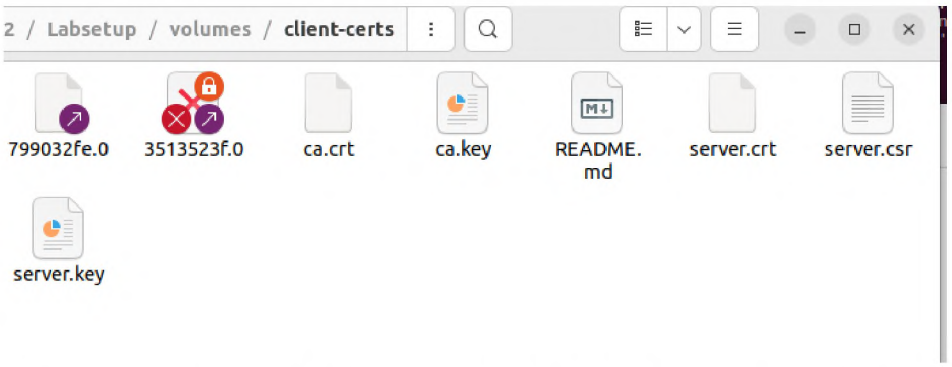

Рисунок 3.12 – Вигляг папки client – certs після додання сертифікатів

Відповідно змінюю файл etc/hosts (рис. 3.13).

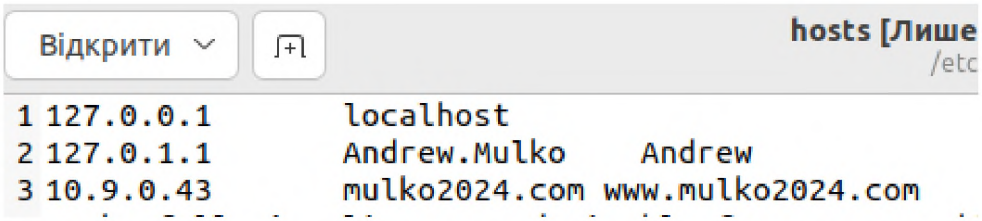

Рисунок 3.13 – Змінення налаштування файлу hosts

Запускаю server.py та виконую TLS – handshake (рис. 3.14).

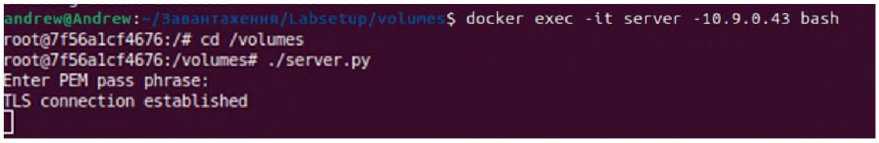

Рисунок 3.14 – Командний рядок після виконання команди server.py

3.5 Тестування серверної программи за допомогою браузера

Браузер бачить сайт як потенційно небезпечний, оскільки я ще не завантажив сертифікат в браузер. Після додання сертифіката ca.crt все запрацювало як потрібно (рис. 3.15).

Рисунок 3.15 – Вигляд сайту після додання сертифікату

Створюю proxy.py (рис. 3.16).

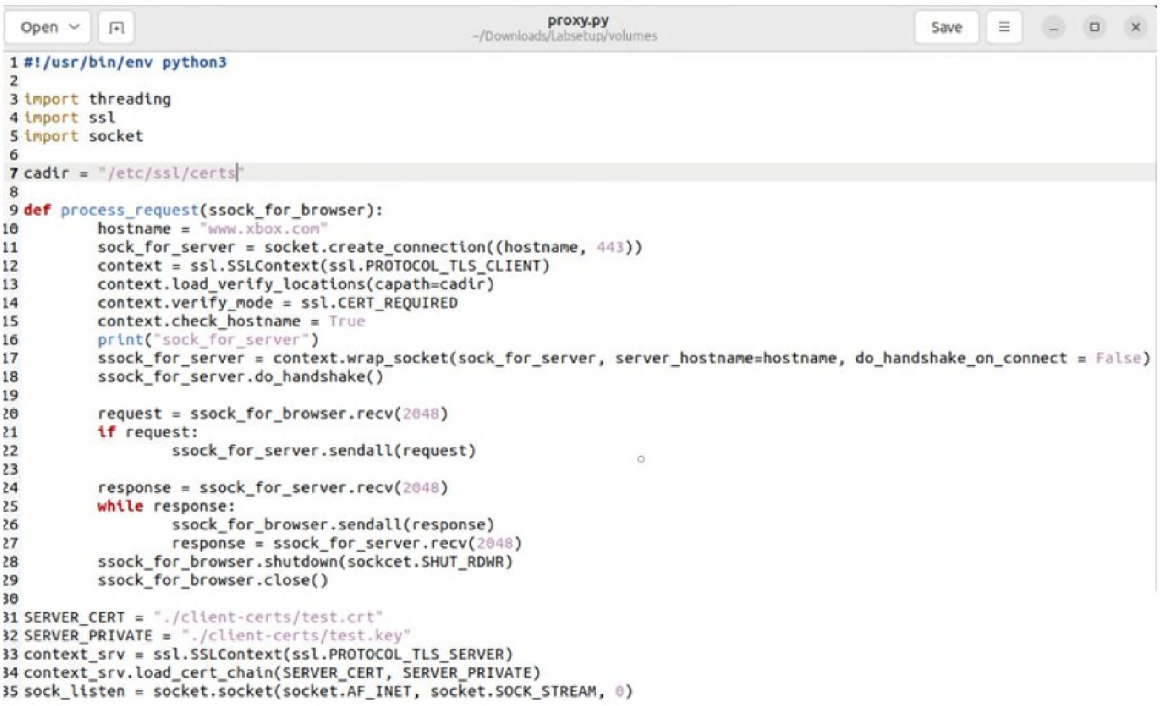

Рисунок 3.16 – Створення та налаштування файлу proxy.py

Налаштовую hosts та створюю сертифікат xbox.com, запускаю hosts.py (рис. 3.17).

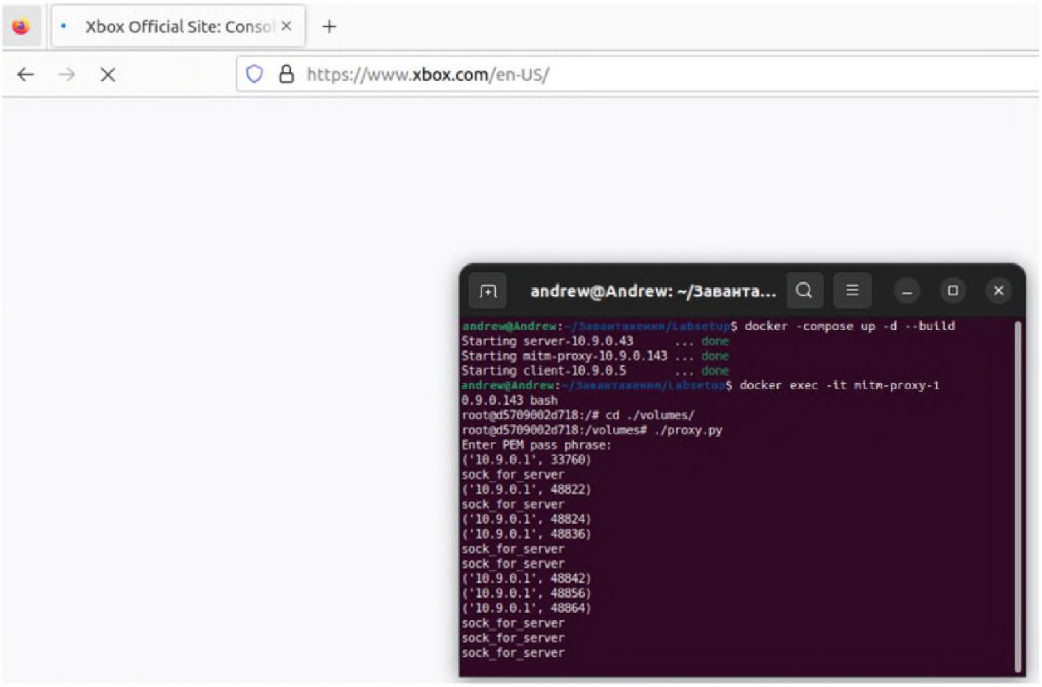

Рисунок 3.17 – Командний рядок після виконання команди hosts.py

Проксі бачe підключення до сайту та в теорії перехвачує деяку інформацію, але повного завантаження сайту не відбувається.

3.6 Оцінка економічної ефективності програмування TLS

Основною метою є досягнення високої економічної ефективності проєкту. Це включає оптимізацію витрат, підвищення якості послуг, що надаються, та забезпечення довгострокової фінансової стабільності.

Основні цілі економічної ефективності:

- оптимізація витрат на навчальний процес;
- підвищення економічної ефективності використання ресурсів;
- підготовка висококваліфікованих фахівців з мінімальними витратами.

Постановка конкретних завдань щодо економічної ефективності:

– розробка інтерактивних модулів для зниження витрат на традиційні навчальні матеріали;

– тестування знань для зменшення витрат на викладацький склад.

Вимоги до економічної цінності та аналіз ринку TLS:

– аналіз поточних потреб ринку для навчання протоколу TLS;

– вивчення економічної цінності існуючих рішень на ринку.

Навчальні заклади потребують економічно ефективних рішень для навчання студентів протоколу TLS, зменшення витрат на обладнання та ресурси, а також підвищення рівня знань студентів у сфері інформаційної безпеки.

Аналіз існуючих рішень на ринку та їх можливостей:

– аналіз існуючих освітніх програм та їх економічної ефективності;

– оцінка можливостей інтеграції та використання протоколу TLS у навчальному процесі.

Законодавчі вимоги та стандарти у сфері захисту інформації:

– вивчення актуальних законодавчих вимог щодо захисту інформації;

– визначення відповідності розроблюваного ПЗ існуючим стандартам безпеки.

Вартість послуг розробників:

– оплата праці розробників: 40 дол. США / год;

– оплата праці тестувальників: 30 дол. США / год.

Середньомісячна оплата за 160 робочих годин:

– розробники: 40 × 160 = 6400 дол. США;

– тестувальники: 30 × 160 = 4800 дол. США.

Таблиця 3.1 – Витрати на розробку ПЗ

Найменування	Сума, долари США	Сума, грн.
Оплата праці розробників	6400	261344
Оплата праці тестувальників	4800	196008
Навчання персоналу	1500	61252
Разом	12700	518605

Таблиця 3.2 – Вартість комп’ютерного обладнання

Найменування	Ціна 1 шт. у доларах США	Ціна 1 шт. в грн	Кількість шт.
Комп’ютер	900	36751	1

Комп'ютерну техніку можна віднести тільки до основних засобів. Розрахунок амортизації лінійним методом провадиться за формулою:

$$A = \frac{\text{С} \times \text{Н}}{100\ \%} \quad (3.1)$$

А – сума амортизаційних відрахувань за рік, грн.;

Н – норма амортизаційних відрахувань, %;

С – середньорічна вартість основних засобів;

Відповідно: 900 × 20 % / 100 % = 180 дол. США.

Таблиця 3.3 – Кошторис витрат

Найменування економічних елементів витрат	Сума, долари США	Сума, грн.
Витрати на оплату праці	1100	44918
Амортизація основних фондів	180	7350
Інші витрати (комунальні послуги)	450	18375
Разом	1730	70644

До розрахунків було прийнято середній за червень 2024 року курс долара по відношенню до гривні: 1 дол. США = 40 грн.

Прогнозовані вигоди від впровадження протоколу TLS:

– зменшення витрат на навчальні матеріали та лабораторні заняття;

– підвищення кваліфікації викладачів;

– зменшення ризиків, пов'язаних з інформаційною безпекою.

Оцінка впливу на якість навчання та підвищення конкурентоспроможності навчального закладу:

– підвищення якості навчання завдяки використанню сучасних технологій;

– залучення більшої кількості студентів завдяки сучасній програмі.

Завдяки впровадженню TLS можна очікувати збільшення кількості студентів на 20 – 30 %.

Розрахунок часу, за який початкові інвестиції у навчанні програмування TLS окупляться за рахунок отриманих вигод. Окупність планується досягти через 2 роки.

Різниця між теперішньою вартістю вигод і витрат протягом півроку складе 5000 дол. США. Ставка дисконту, при якій NPV дорівнює нулю, становить 15 %. Відношення чистого прибутку до інвестицій становить 25 %.

Визначення можливих ризиків:

- зміни ринку;
- технічні проблеми;
- зміни в законодавстві.

Розробка планів на випадок виникнення ризиків:

- оперативна адаптація до змін ринку;
- регулярне технічне обслуговування та оновлення;
- підготовка до змін у законодавстві.

Тестування навчання протоколу TLS на реальних групах студентів буде проведено протягом одного семестру.

Після завершення тестування будуть проведені оцінки результатів, виявлення недоліків та їх усунення. Проєкт виявився економічно ефективним і доцільним для впровадження в навчальний процес. Отримані дані свідчать про високу економічну ефективність проєкту.

Інвестування в розробку та впровадження протоколу TLS в навчальний процес є доцільним.

ВИСНОВКИ

Після ретельного аналізу безпеки транспортного рівня Transport Layer Security (TLS), створення сертифікатів X.509 та розроблення програмного забезпечення для аналізу TLS, можна зробити наступні висновки:

Протокол TLS відіграє ключову роль у забезпеченні безпечного зв'язку та захисту конфіденційних даних в Інтернеті. Його ретельне вивчення та правильна реалізація є критично важливими для підтримки належного рівня безпеки в мережевих додатках.

Сертифікати X.509 з розширеннями альтернативного імені суб'єкта (SAN) є необхідними для забезпечення автентичності та цілісності TLS – з'єднань. Створення та належне управління такими сертифікатами є важливим завданням для забезпечення безпеки.

Розроблене програмне забезпечення для аналізу TLS – з'єднань може ефективно виявляти потенційні вразливості та недоліки в реалізації TLS. Воно дозволяє перевіряти дійсність сертифікатів, аналізувати використовувані шифри та протоколи, а також виявляти застарілі або вразливі версії TLS.

Використання такого програмного забезпечення для регулярного аналізу та моніторингу TLS – з'єднань може значно підвищити рівень безпеки мережевих додатків та захистити їх від можливих атак та витоку даних.

Постійний моніторинг нових вразливостей та своєчасне оновлення програмного забезпечення та бібліотек для роботи з TLS є критично важливими для підтримки належного рівня безпеки.

Загалом, проведений аналіз та розроблене програмне забезпечення забезпечують комплексний підхід до вирішення питань безпеки транспортного рівня та захисту конфіденційних даних під час передачі через Інтернет. Це допоможе підвищити загальний рівень безпеки мережевих додатків та запобігти витоку чутливої інформації.

При виконанні кваліфікаційної роботи було виконано аналіз та розробку TLS на операційній системі Linux Ubuntu.

При підготовці розділу 2, було зібрано та сформульовано функціональні вимоги до розробки, до архітектури та функціоналу.

У третьому розділі було зосереджено увагу на виборі найбільш придатного набору інструментів для розробки Transport Layer Security (TLS). Після ретельного аналізу наявних технологій та фреймворків, було обрано оптимальний набір інструментів, який найповніше відповідає вимогам та специфіці проєкту TLS.

Після вибору інструментів був проведений ґрунтовний аналіз та розпочато безпосередню розробку TLS. У процесі розробки було враховано всі виявлені вимоги, специфіку предметної області та найкращі практики програмування, що забезпечило створення якісного та надійного рішення для забезпечення захищеного з'єднання між клієнтами та серверами. Проведено розрахунок вартості виконаних робіт.