НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	Завідувач кафедри
Інформаційних технологій	Комп'ютерних систем, мереж та кібербезпеки
_____ Болбот І.М., д.тех.н, проф.	_____ Касаткін Д.Ю., к. пед.н., доц.
підпис ПІБ, вчене звання і ступінь	підпис ПІБ, вчене звання і ступінь
«___»_____ 2025 р.	«___»_____ 2025 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

На тему: «Комп'ютерна система криптографічного захисту інформації»

Спеціальність F7 «Комп’ютерна інженерія»

Гарант освітньої програми:_____

Керівник дипломного проекту:_____ / Кулініч О.М. /
підпис ПІБ

Виконав:_____ / Плюта П.М./
підпис ПІБ

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«ЗАТВЕРДЖУЮ»
завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
_____ / Касаткін Д.Ю., к.п.н., доц. /
підпис ПІБ, вчене звання і ступінь
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ БАКАЛАВРСЬКОЇ СТУДЕНТУ

Плюта Павло Миколайович
(прізвище, ім'я, по батькові)
Спеціальність (напрям підготовки): Комп'ютерна інженерія_____

Тема кваліфікаційної бакалаврської роботи: «Комп'ютерна система криптографічного захисту інформації»_____
_____
_____
затверджена наказом ректора НУБіП України від "___" _____202_р. №____
Термін подання завершеної роботи на кафедру _____
Вихідні дані до кваліфікаційної бакалаврської роботи _____
_____
_____
_____
_____
_____
Перелік питань, що підлягають розробці:
1. _____
2. _____
3. _____
Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_____"_____2024 р.

Керівник бакалаврської роботи _____ _Кулініч О.М. д.т.н._
(підпис) (прізвище та ініціали)
Завдання прийняв до виконання _____ _____Плюта П.М._____
(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області	04.03.2025 р.	Виконано
2	Проектування системи	15.04.2025 р.	Виконано
3	Реалізація системи	10.04.2025 р.	Виконано
4	Тестування системи	01.05.2025 р.	Виконано
5	Оформлення пояснювальної записки	13.05.2025 р.	Виконано
6	Оформлення графічного матеріалу	13.05.2025 р.	Виконано

Студент _____ **Павло ПЛЮТА**

(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **Олег КУЛІНІЧ**

(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 61 сторінка, 19 рисунків, 14 таблиць, 1 лістинг, 2 додатки, 24 джерела.

Ключові слова: комп'ютерна система, криптографічний захист інформації, гібридна криптосистема, AES-256, RSA-4096, симетричне шифрування, асиметричне шифрування, обмін ключами, графічний інтерфейс користувача, Tkinter, Python, захист даних, шифрування файлів.

Об'єкт аналізу – процес захисту інформації в комп'ютерних системах.

Мета роботи – розроблення настільної комп'ютерної системи, яка забезпечує криптографічний захист інформації за гібридним підходом AES-256 + RSA-4096.

Проект складається з трьох розділів.

Перший розділ присвячено загальним поняттям криптографії, опису основних криптографічних примітивів та їхніх характеристик.

У другому розділі проаналізовано криптографічні протоколи захисту інформації в мережі Інтернет, зокрема SSL/TLS, IPsec, VPN-технології та інші.

У третьому розділі проведено дослідження безпеки веб-сервера для різних версій протоколів SSL та TLS, виявлено вразливості та запропоновано заходи з їх усунення.

У результаті було розроблено програмний комплекс для шифрування й дешифрування файлів та текстових повідомлень, який може бути інтегрований у корпоративні або персональні інформаційні системи.

Зміст

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

AES	–	Advanced Encryption Standard
RSA	–	Rivest Shamir Adleman
SSL	–	Secure Sockets Layer
SSH	–	Secure Shell
IPsec	–	Internet protocol Security
L2TP	–	Layer 2 Tunneling Protocol
IKE	–	Internet Key Exchange
IKEv2	–	Internet Key Exchange version 2
MOBIKE	–	Mobility and Multihoming IKEv2
AH	–	Authentication Header
ESP	–	Encapsulating Security Payload
S/MIME	–	Securt/Multipurpose Internet Mail Extensions
OpenPGP	–	Open Pretty Good Privacy
ARP	–	Address Resolution Protocol
DNS	–	Domain Name System
HTTP	–	HyperText Transfer Protocol
HTTPS	–	HTTP Secure
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
VPN	–	Virtual Private Network
API	–	Application Programming Interface
TLS	–	Transport Layer Security
ПК	–	Персональний комп’ютер
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення

ВСТУП

У цифрову епоху безпека даних в Інтернеті виходить на перший план. Щодня між домашніми комп'ютерами й віддаленими серверами циркулюють гігабайти інформації, часто — конфіденційної. Банківські платежі, робоче листування, особисті повідомлення та інші онлайн-дії стали нашою буденністю, а разом із тим — привабливою мішенню для кіберзлочинців.

Сучасні виклики спонукали до появи та постійного вдосконалення криптографічних протоколів, покликаних гарантувати безпеку даних в Інтернеті. Завдяки різноманітним методам шифрування ці протоколи створюють надійні умови для захищеного передавання інформації й роблять її недосяжною для зловмисників. Однак завдання захисту даних у мережі стає дедалі складнішим, тож воно потребує безперервних досліджень і впровадження найновіших технологічних рішень.

Актуальність цієї дипломної роботи зумовлена потребою глибоко дослідити та оцінити різні криптографічні протоколи, зокрема SSL і TLS, аби підвищити рівень захисту вебсерверів. Головна мета дослідження — проаналізувати, зіставити та дати оцінку протоколам, що забезпечують безпеку інформації в Інтернеті, щоб визначити їхні переваги, недоліки та можливі вразливості. Об'єктом роботи виступають криптографічні протоколи мережевого захисту, а предметом — їхні алгоритми, а також методи виявлення й усунення слабких місць.

Попри значний арсенал уже існуючих криптографічних протоколів, дослідники не зупиняються: стрімкий розвиток квантових обчислень може кардинально «зламати» чинні схеми шифрування. Сьогодні створюються й тестуються нові алгоритми та протоколи, здатні протистояти ймовірним квантовим атакам. Щоб рухатися далі й підсилювати захист, важливо тверезо оцінювати актуальний стан цих рішень, розуміти їхній запас міцності й помічати слабкі місця.

Таким чином, виконане дослідження має відчутне як теоретичне, так і прикладне значення. У роботі ґрунтовно проаналізовано найпоширеніші криптографічні протоколи, які застосовують для захисту даних під час передавання мережею. Детальне вивчення їхньої структури та впливу на рівень безпеки допомагає краще налаштовувати й удосконалювати процес обміну інформацією.

У роботі значний акцент зроблено на перевірку безпечності системи. Результати свідчать: коли криптопротоколи впроваджено та налаштовано коректно, захист даних під час передавання через Інтернет відчутно посилюється, а сам процес стає більш надійним.

РОЗДІЛ 1 ЗАГАЛЬНІ ПОНЯТТЯ КРИПТОГРАФІЇ

1.1 Загальні криптографічні поняття

Криптографія—це галузь, що застосовує математичні методи для збереження та підтвердження конфіденційності, справжності й цілісності даних. Її головна ціль — убезпечити інформацію від стороннього доступу, несанкціонованих змін і неправомірного використання.

Сьогодні криптографічні методи лежать в основі роботи цілої низки сфер — від кібербезпеки й онлайн-торгівлі до банківських сервісів та мобільних комунікацій. Саме вони захищають інформацію, що мандрує мережами, зберігається на носіях чи обмінюється між користувачами, убезпечуючи її від стороннього втручання.

Основні принципи криптографії включають:

Конфіденційність передбачає, що жодна інформація не потрапляє до рук неавторизованих осіб. Перед обміном даними відправник і одержувач спочатку проходять процедури ідентифікації та автентифікації, щоб підтвердити свою особу. Цілісність даних гарантує, що інформація не зазнає змін або не буде переміщена під час передачі. Забезпечення невідмовності полягає в тому, що жодна із сторін не може відмовитися від створення чи відправки повідомлення, що дозволяє відстежити й підтвердити всі транзакції.Нижче наведено схему, яка ілюструє основні криптографічні примітиви та їх зв'язок один з одним (див. рисунок 1.1).

У криптографії шифрування означає перетворення відкритого тексту в зашифровану форму за допомогою певного алгоритму та ключа, а розшифрування — це оборотний процес, при якому зашифроване повідомлення повертається до вихідного тексту з використанням відповідного ключа. Дешифрування виконує таке ж відновлення без застосування ключа. Ключ є параметром, необхідним для шифрування і розшифрування: у випадку симетричної криптографії використовується один і той самий секретний ключ,

тоді як в асиметричній кожний із ключів (приватний і публічний) відповідає за свою операцію. Під алгоритмом шифрування розуміють математичну процедуру, що задає правила перетворення відкритого повідомлення в зашифроване. Автентифікація — це перевірка справжності особи чи сутності, що стверджує свою ідентичність. Хешування ж полягає в перетворенні довільних даних на фіксовану довжину хеш-значення за допомогою спеціальної хеш-функції.

Щоб створити стійку криптографічну систему, базові примітиви часто застосовують багаторазово, використовуючи операції підстановки, перестановки, циклічного зсуву та побітового XOR. Основними криптографічними примітивами є симетричне шифрування, асиметричне шифрування, цифровий підпис та хеш-функції.

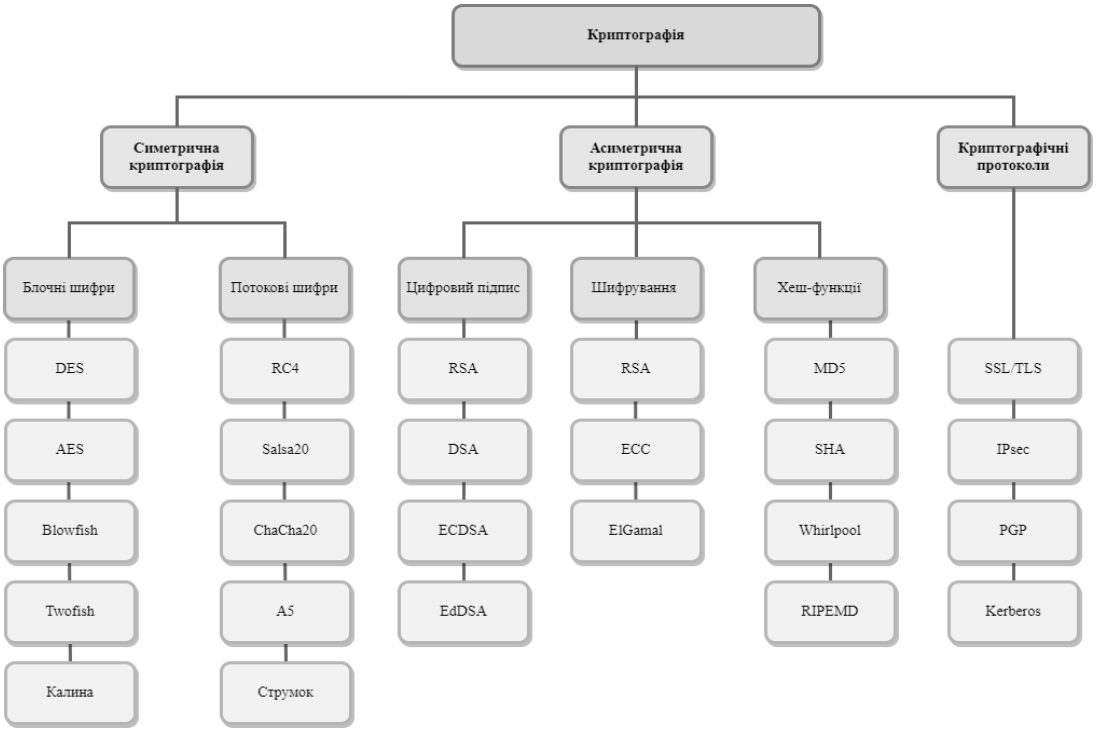

Рисунок 1.1 – Основні криптографічні примітиви

Симетричне шифрування належить до базових засобів криптографії: той самий секретний ключ застосовується як під час зашифрування, так і під час розшифрування даних (див. рис. 1.2). До класичних прикладів таких

алгоритмів належать AES і DES. Рішення вирізняється високою швидкодією, однак вимагає надійного каналу для обміну ключем між учасниками зв’язку.

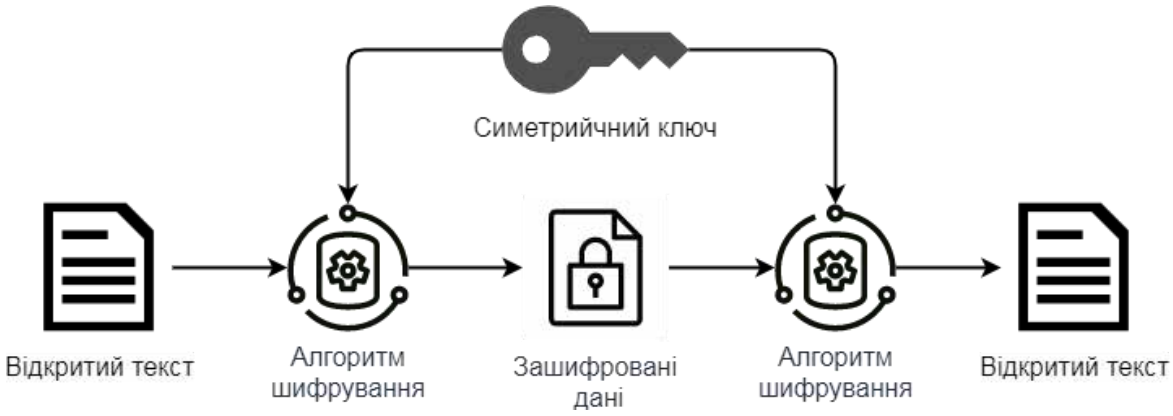

Рисунок 1.2 – Схема симетричного шифрування

Асиметричне шифрування спирається на пару різних ключів — публічний і приватний (див. рис. 1.3). Шифрування виконують публічним ключем, а розшифрування можливе лише за допомогою відповідного приватного. Найпоширеніші алгоритми цього типу — RSA та ECC. Такий підхід гарантує безпечний обмін ключами, проте працює повільніше, коли потрібно обробити великі масиви даних.

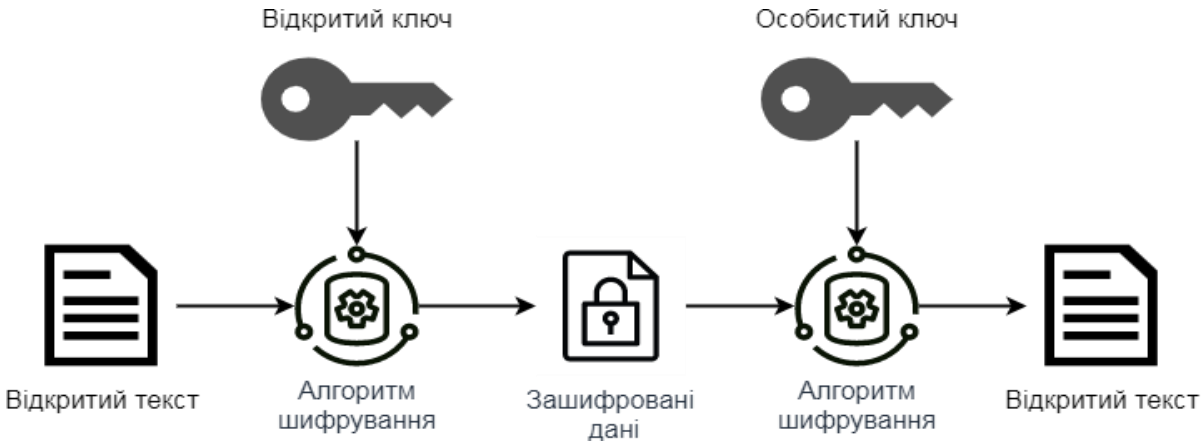

Рисунок 1.3 – Схема асиметричного шифрування

Симетричні та асиметричні алгоритми — два фундаментальні підходи до шифрування, що забезпечують захист даних у цифровому просторі. Кожен із них має власні сильні сторони й недоліки, тому вибір між ними визначається конкретними вимогами до безпеки й зручності використання. У таблиці 1.1 подано зіставлення цих двох методів за ключовими критеріями: принцип роботи, спосіб оброблення інформації, механізм обміну ключами та сфери застосування. Також наведено приклади типових алгоритмів для кожної групи. Такий огляд допомагає чітко побачити особливості та специфіку обох підходів і об’єктивно оцінити їхні можливості й обмеження.

Таблиця 1.1 – Порівняння симетричного та асиметричного шифрування за ключовими аспектами

Аспект	**Симетричне шифрування**	**Асиметричне шифрування**
Принцип шифрування	Використовується один і той же ключ для шифрування та розшифрування даних	Використовуються два ключі: приватний і публічний ключі
Обробка даних	Швидка та ефективна	Може бути повільною залежно від розміру даних
Обмін ключами	Вимагає безпечного обміну однаковим ключем між відправником і отримувачем	Використовується публічний ключ для шифрування та приватний ключ для розшифрування
Застосування	Захист великих обсягів даних, шифрування файлів та дисків	Електронна підписка, аутентифікація, обмін ключами
Приклади алгоритмів	AES, DES, 3DES	RSA, ECC, ElGamal

Наступна таблиця 1.2 надає огляд деяких популярних алгоритмів шифрування, разом з їх коротким описом. Ці алгоритми розроблені для різних цілей і мають різні характеристики щодо безпеки, швидкодії та масштабованості.

Таблиця 1.2 – Популярні алгоритми шифрування

Алгоритм шифрування	Опис
AES (Advanced Encryption Standard)	Симетричний блочний алгоритм шифрування, що використовується для захисту конфіденційності даних
RSA (Rivest-Shamir-Adleman)	Асиметричний алгоритм шифрування, який використовується для шифрування та цифрових підписів
ECC (Elliptic Curve Cryptography)	Асиметричний алгоритм шифрування, оснований на математичних властивостях еліптичних кривих
Blowfish	Симетричний блочний алгоритм шифрування з високою швидкодією
3DES (Triple Data Encryption Standard)	Симетричний блочний алгоритм шифрування, що застосовується у трьох послідовних раундах

Цифровий підпис — це електронний еквівалент звичайного автографа, який гарантує автентичність і цілісність переданих даних. Він формується за допомогою приватного ключа, а перевіряється відповідним публічним ключем. Завдяки цьому механізму можна достовірно підтвердити авторство повідомлення й запобігти відмові від нього. До найпоширеніших алгоритмів цифрового підпису належать RSA, DSA, ECDSA й EdDSA. Стислий огляд цих алгоритмів наведено в таблиці 1.3.

Таблиця 1.3 – Популярні алгоритми цифрового підпису

Тип цифрового підпису	Опис
RSA (Rivest-Shamir-Adleman)	Асиметричний алгоритм, який використовується для цифрових підписів та шифрування даних
DSA (Digital Signature Algorithm)	Стандартний асиметричний алгоритм для цифрових підписів
ECDSA (Elliptic Curve Digital Signature Algorithm)	Асиметричний алгоритм, оснований на еліптичних кривих, використовується для цифрових підписів
EdDSA (Edwards-curve Digital Signature Algorithm)	Сучасний асиметричний алгоритм для цифрових підписів, який базується на кривих Едвардса

Хеш-функції потрібні, щоб перетворювати повідомлення будь-якої довжини на коротке хеш-значення фіксованого розміру. Щоби вважатися надійними, вони мусять бути односторонніми, тобто за готовим хешем неможливо відновити початкові дані, а також стійкими до колізій, коли практично нереально знайти дві різні вхідні стрічки з однаковим результатом. Найвідомішими прикладами залишаються SHA (Secure Hash Algorithm) і MD5 (Message Digest 5) — класичні стандарти, які й досі часто зустрічаються в системах контролю цілісності та підпису. Короткі характеристики цих і кількох інших популярних хеш-функцій зібрано в таблиці 1.4.

Таблиця 1.4 – Опис найбільш відомих хеш-функцій

Тип хеш-функції	Опис
MD5 (Message Digest Algorithm 5)	Широко використовуваний криптографічний хеш-алгоритм, який створює 128-бітний хеш-значення. Зараз вважається небезпечним через вразливості до атак.
SHA-1 (Secure Hash Algorithm 1)	Криптографічний хеш-алгоритм, який створює 160-бітний хеш-значення. Вважається небезпечним через здатність до колізій.
SHA-2 (Secure Hash Algorithm 2)	Сімейство хеш-алгоритмів з різними довжинами хеш- значення (224, 256, 384, 512, 512/224, 512/256 біт). Використовується в багатьох застосунках безпеки.
SHA-3 (Secure Hash Algorithm 3)	Останнє покоління Secure Hash Algorithm. Переможець конкурсу NIST на криптографічний хеш-алгоритм.

Розібравшись із різновидами хеш-функцій, доцільно поверхово зіставити їх із цифровим підписом, аби краще зрозуміти, у яких ситуаціях кожен інструмент є доречним. У таблиці 1.5 наведено стислий огляд їхніх основних властивостей і того, наскільки ефективно вони виконують свої завдання.

Таблиця 1.5 – Порівняння хеш-функцій і цифрового підпису

Аспект	Хеш-функція	Цифровий підпис
Використання ключів	Не потребує використання ключів	Приватний ключ для підписування та публічний ключ для перевірки
Конфіденційність	Ні	Ні
Цілісність	Так	Так
Застосування	Верифікація цілісності даних, порівняння даних	Автентифікація, цифровий підпис, безвідмовність даних
Алгоритми	MD5, SHA-1, SHA-256	RSA, DSA, ECDSA

Криптографічні протоколи — це узгоджені правила, які визначають, як саме слід застосовувати криптографічні методи, щоб у конкретній ситуації гарантувати безпеку. Вони охоплюють процедури автентифікації, шифрування, накладання цифрового підпису й інші захисні операції.

У таблиці 1.6 подано різні сфери застосування криптографічних протоколів — захист даних, мережеву безпеку, шифрування електронної пошти та захищені з'єднання. Для кожної з них зазначено найпоширеніший протокол і стислий опис його головних характеристик.

Таблиця 1.6 –Ключові аспекти криптографічних протоколів

Аспект	**Криптографічний протокол**	**Основні характеристики**
Захист даних	SSL/TLS	Шифрування, автентифікація, цифровий підпис
Безпека мереж	IPsec	Шифрування на рівні мережі
Захист електронної пошти	PGP, S/MIME	Шифрування, автентифікація, цифровий підпис
Захист з'єднання	SSH	Шифрування, автентифікація

Плануючи криптозахист даних, слід орієнтуватися на чинні галузеві стандарти й рекомендації. Головне — використовувати перевірені, стійкі алгоритми та надійно зберігати ключі, адже саме вони є «серцем» будь-якої шифрувальної системи. Конкретний набір методів добирають з огляду на вимоги й ризики конкретної програми чи сервісу, постійно звіряючись із сучасними практиками інформаційної безпеки.

1.2 Порівняння криптографічних примітивів

У криптографії є ціла низка базових елементів, що відповідають за безпеку та збереження таємниці даних. Щоб правильно обрати, яким саме способом захищати конкретну систему, треба добре розуміти, у чому полягають відмінності між цими примітивами.

Симетричне шифрування — це базовий криптографічний прийом, у якому той самий секретний ключ служить і для зашифрування, і для розшифрування даних. Працює воно дуже швидко й не навантажує систему, але є нюанс: цей ключ треба передати іншій стороні по-справжньому безпечним каналом.

Асиметричне шифрування працює з двома різними ключами: відкритим і приватним. Відкритий ключ віддають усім охочим, щоб вони могли зашифрувати для вас повідомлення, а розшифрувати його здатен лише власник відповідного приватного ключа. Плюс у тому, що немає потреби передавати секретний ключ іншим, тож ризик «перехоплення» майже зникає. Мінус — такі операції важчі для процесора й займають більше часу, ніж симетричне шифрування.

Цифровий підпис дає змогу переконатися, що повідомлення справді надіслав заявлений автор і що його зміст ніхто не перекрутив. Для його формування дані спершу хешуються, а потім цей хеш шифрується приватним ключем власника. Перевіряють підпис зворотним способом — розшифровуючи його відкритим ключем. Такий підхід гарантує незмінність даних та унеможливлює відмову від авторства.

Хеш-функції, у свою чергу, «стискають» будь-який обсяг інформації до рядка фіксованої довжини — хешу. Такий відбиток слугує для перевірки цілісності повідомлень і файлів, а ще дає змогу створювати унікальні ідентифікатори. Важлива риса: це перетворення одностороннє, тож із самого хеша вихідні дані відновити неможливо.

У таблиці 1.7 зведено короткі характеристики основних криптографічних «цеглинок», на яких зводять складні системи захисту. Який

саме прийом використовувати, вирішують з огляду на завдання, обмеження й ризики конкретної системи. Уважно проаналізувавши властивості кожного примітиву, можна вибрати протоколи, що найкраще відповідають потрібному рівню безпеки, і побудувати захисну стратегію, оптимальну саме під ваші умови.

Таблиця 1.7 – Порівняння криптографічних примітивів

Аспект	**Симетричне шифрування**	**Асиметричне шифрування**	**Цифровий підпис**	**Хеш-функція**
Основний принцип	Використовується один і той же ключ для шифрування та розшифрування даних	Використовується публічний та приватний ключі для шифрування та розшифрування даних	Використовується приватний ключ для підписування даних та публічний ключ для перевірки підпису	Генерація унікального хешу для вхідних даних, який використовується для верифікації цілісності
Ключі	Вимагає обмін та зберігання одного і того ж ключа між відправником і отримувачем	Використовує два ключі: приватний (зберігається в таємниці) та публічний (розповсюджується)	Використовується приватний ключ для підписування та перевірки підпису	Ключі не використовуються
Підтримка конфіденційності	Так	Так	Ні	Ні
Підтримка цілісності	Ні	Так	Так	Так
Витрати обчислювальних ресурсів	Низькі	Високі	Високі	Залежить від протоколу та алгоритму шифрування
Використання у реальному часі	Так	Так	Так	Так
Основне застосування	Шифрування даних на місці та під час передачі	Шифрування, аутентифікація, цифрові підписи	Підтвердження авторства, цілісності та безвідмовності	Верифікація цілісності, створення відбитків даних
Приклади алгоритмів	AES, DES, 3DES	RSA, ECC	RSA, ECDSA	MD5, SHA-1, SHA-2, SHA-3

У таблиці подано стислий огляд основних властивостей симетричного та асиметричного шифрування, цифрового підпису й хеш-функцій.

РОЗДІЛ 2. КРИПТОГРАФІЧНІ ПРОТОКОЛИ ЗАХИСТУ ІНФОРМАЦІЇ В МЕРЕЖІ ІНТЕРНЕТ

2.1 Криптографічні протоколи

2.1.1 Огляд криптографічних протоколів

Криптографічний протокол — це узгоджений набір правил і процедур, що визначає, як саме учасники чи системи мають обмінюватися даними, аби зберегти їх у безпеці. У таких протоколах комбінують різні криптографічні примітиви та алгоритми, щоби захистити інформацію від стороннього доступу, підроблення чи несанкціонованих змін.

У криптопротоколах чітко прописано, хто й у якій послідовності що робить: спершу все ініціалізують, потім сторони впізнають одна одну, домовляються про ключі, далі шифрують і розшифровують дані, перевіряють, чи нічого не змінилося, — і так крок за кроком. Такий сценарій гарантує, що інформація під час обміну лишається конфіденційною, не спотворюється й справді належить тому, хто її відправив.

У протоколі «учасник» — це будь-хто, хто реально щось робить: людина, програма чи навіть компанія. Учасником вважають лише того, хто виконує активні дії, передбачені правилом протоколу.

Сам протокол можна уявити як послідовність ходів. Під час одного ходу працює лише один учасник: спершу він отримує повідомлення, а потім надсилає своє. Кожен хід складається з кількох кроків — це окремі, завершені дії.

Наприклад:

- генерація нового (випадкового) значення;
- обчислення значень функції;
- перевірка ключів, сертифікатів, підписів тощо;
- прийом, відправка повідомлень.

Рисунок 2.1 – Схематичне зображення циклу дій учасника протоколу

Серед найуживаніших криптографічних протоколів можна назвати SSL/TLS, які захищають з'єднання з веб-сайтами; схеми автентифікації на кшталт OAuth і OIDC; а також протоколи обміну ключами, наприклад Диффі-Гелмана. Ці рішення широко застосовують у різних сферах — від захисту електронної пошти та мобільних застосунків до мережевих сервісів.

Завдання криптографічних протоколів — гарантувати безпеку даних у відкритих чи ненадійних мережах, де завжди є ризик атак і несанкціонованого доступу. Щоб інформаційні системи справді лишалися конфіденційними, цілісними й доступними, вкрай важливо коректно спроєктувати, грамотно реалізувати й правильно застосовувати такі протоколи.

2.1.2 Класифікація криптографічних протоколів

Універсальної класифікації криптографічних протоколів поки що не сформовано, проте їх можна групувати за різними ознаками — за призначенням, кількістю учасників, характером взаємодії та іншими параметрами.

Втім, спираючись на окремі ознаки й характеристики (див. рис. 2.2), протоколи можна згрупувати у кілька різних класів.

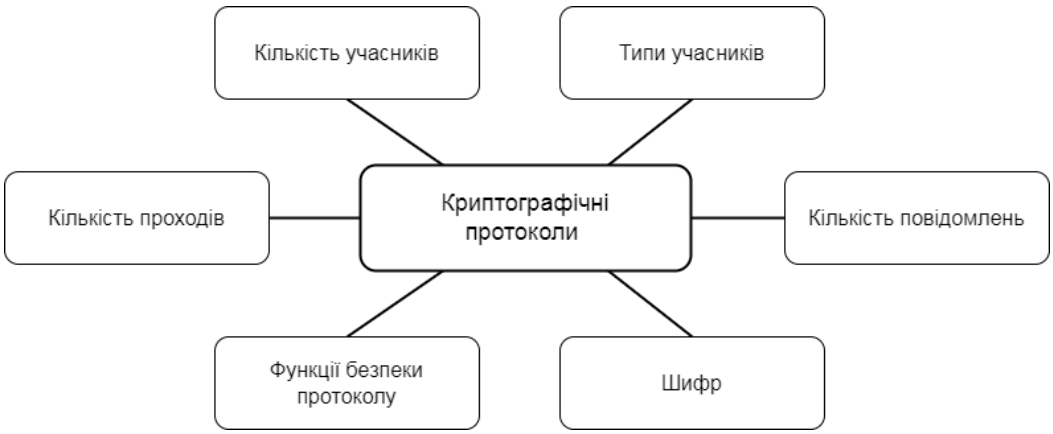

Рис. 2.2 – Основні критерії класифікації криптографічних протоколів.

За кількістю учасників протоколи можуть поділятись на:

- двосторонні;
- тристоронні;
- багатосторонні.

За кількістю повідомлень, що передаються, протоколи можуть класифікуватись наступним чином:

- інтерактивний (взаємний обмін повідомленнями);
- неінтерактивний (одноразова передача повідомлень між учасниками).

У криптографічних протоколах можуть використовуватись:

- симетричні шифри;
- асиметричні шифри.

Захищені системи й протоколи відповідають за різні аспекти безпеки, які часто зводять до протистояння певним типам атак. Найповніший та найактуальніший список таких завдань із поясненнями подано в документації проєкту AVISPA, що узагальнює матеріали з численних RFC IETF. Прийнято вважати, що для кожного конкретного протоколу можна формально довести або спростувати досягнення цих цілей.

2.2 Криптографічні протоколи захисту інформації у мережі Інтернет

Інтернет — це децентралізована мережа, у якій інформація рухається через безліч вузлів і маршрутизаторів, що не завжди піддаються контролю. Тож захист даних треба організовувати в непередбачуваних і ненадійних умовах, запобігаючи їх перехопленню, підміні чи будь-якому втручанню під час передавання. Щодня мережею проходять колосальні обсяги трафіку — від веб-сторінок і електронної пошти до файлів, потокового відео й аудіо. Тому системи безпеки мають працювати стабільно й ефективно навіть за пікового навантаження та великих потоків даних.

В Інтернеті одночасно функціонує безліч протоколів, пристроїв, операційних систем і програмних платформ, що постійно взаємодіють між собою. Щоб дані залишалися захищеними за будь-яких умов, засоби безпеки повинні ґрунтуватися на спільних стандартах і коректно працювати у всіх цих середовищах та на будь-яких пристроях.

Після того, як Інтернет набув комерційного характеру, криптографія природно стала його органічною складовою. Сьогодні шифрувальні протоколи міцно вписані у мережеві стандарти, адже саме вони забезпечують захищене передавання інформації.

Першою «вітриною» криптографії в Інтернеті стала електронна пошта: її почали шифрувати, аби гарантувати конфіденційність і цілісність повідомлень під час передавання.

Для захисту електронної кореспонденції застосовували S/MIME і PGP: ці протоколи шифрують повідомлення та додають цифрові підписи, зберігаючи їхню конфіденційність і підтверджуючи справжність.

Щоб захистити трафік між веб-сервером і браузером, розробили протокол SSL. Вбудований у популярні оглядачі, він шифрував усю конфіденційну інформацію, яку користувачі передавали через веб-сайти.

SSL і його сучасна версія — TLS — слугують для створення захищеного каналу між клієнтом і сервером. Ці протоколи оберігають трафік веб-сторінок, електронної пошти, платіжних транзакцій та інших онлайн-сервісів. Використовуючи шифрування й перевірку цифрових сертифікатів, SSL/TLS гарантує конфіденційність, цілісність і автентичність передаваних даних.

Для захисту трафіку на рівні IP з'явилися нові криптографічні протоколи — PPTP, L2TP та, передусім, IPsec. Їх найчастіше застосовують під час створення віртуальних приватних мереж (VPN), адже вони охоплюють більшу частину мережевого обміну. Особливо IPsec, який працює безпосередньо на мережевому рівні: він шифрує IP-пакети, перевіряє їхню цілісність і автентичність, тим самим забезпечуючи надійний захист даних у транзиті.

Окрім уже згаданих рішень, для захищених віддалених сесій активно застосовують протокол SSH. Його призначення — гарантувати безпечний доступ до віддалених машин і виконання на них команд, шифруючи весь обмін і перевіряючи автентичність користувача. Завдяки цьому сторонні особи не можуть перехопити чи підмінити дані.

Сьогодні криптографічні протоколи залишаються опорою безпеки інтернет-трафіку та охорони приватних даних користувачів. Технології не стоять на місці, разом із ними з'являються і витонченіші способи атак, тож сучасна криптографія постійно перебуває в гонитві озброєнь: рішення слід безупинно оновлювати й удосконалювати, аби не дати новим загрозам жодного шансу.

Сьогодні ключова тенденція розвитку криптографії — перехід на потужніші схеми шифрування. Класичні алгоритми на кшталт RSA чи DES поступово поступаються місцем сучаснішим рішенням: зокрема, еліптичним кривим (ECC) у світі відкритого ключа та передовим блочним шифрам, таким як AES. Завдяки цьому підвищується і загальний рівень захисту, і стійкість систем до криптоаналізу.

Інша важлива тенденція полягає в тому, що шифрувальні механізми поступово охоплюють всі «поверхи» інтернет-інфраструктури. Поряд із добре

знайомими SSL/TLS для захисту веб-сеансів дедалі ширше застосовують нові рішення: DNSSEC гарантує достовірність та цілісність записів DNS, HTTPS робить протокол HTTP безпечним «за замовчуванням», IPsec шифрує трафік уже на рівні IP-пакетів, а схеми OAuth і OIDC відповідають за надійну автентифікацію користувачів і сервісів. Завдяки цьому безпека вибудовується не точково, а на всіх ключових рівнях мережі.

DNSSEC — це розширення класичного DNS, покликане захистити систему доменних імен від підміни даних і фальшивих відповідей серверів. Завдяки йому цілісність записів і справжність їхнього джерела перевіряються криптографічно, тож користувачі можуть без побоювань довіряти отриманим DNS-відповідям.

OAuth — це механізм автентифікації й авторизації, який дає змогу стороннім застосункам отримувати обмежений доступ до захищених ресурсів від імені користувача, не вимагаючи передачі його пароля. Його найчастіше використовують для входу через соціальні мережі та під час роботи з різними API.

OpenID Connect (OIDC) виступає надбудовою над OAuth і відповідає саме за автентифікацію. Він дає застосункам можливість безпечно звернутися до служби ідентифікації, отримати підтверджені відомості про користувача й передати їх назад у програму. У підсумку особисті дані залишаються захищеними протягом усього процесу обміну.

В цілому, сучасний стан використання криптографічних протоколів у мережі Інтернет відображає постійний розвиток та адаптацію технологій до нових проблем та вимог безпеки. Використання сильних алгоритмів, розширення сфери застосування протоколів та урахування балансу між безпекою та зручністю використання є ключовими аспектами у забезпеченні безпеки в Інтернеті.

2.3 Технологія VBN як приклад використання криптографічних протоколів в мережі Інтернет

VPN - це технологія, що створює безпечне з'єднання через небезпечну мережу, таку як Інтернет. Вона дозволяє користувачам отримувати та відправляти

Віртуальна приватна мережа (VPN) вибудовує захищений «тунель» поверх недовіреної мережі — наприклад, Інтернету. Завдяки цьому користувачі можуть обмінюватися даними через публічну інфраструктуру так, ніби їхні пристрої під'єднані безпосередньо до приватної мережі (див. рисунок 2.3).

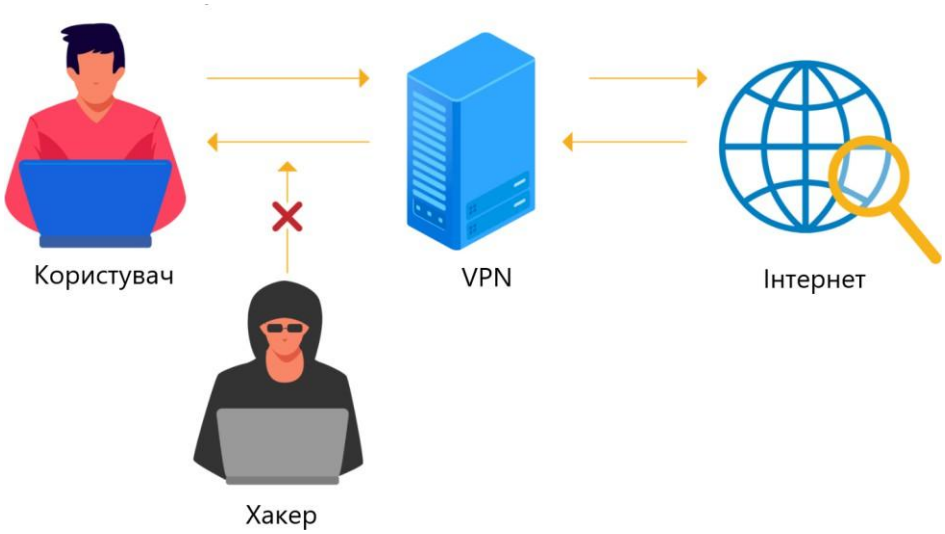

Рисунок 2.3 – Схематичне зображення використання технології VPN

Для реалізації цих сценаріїв існують різні види протоколів VPN: для зв'язку, для шифрування трафіку та інші, і вже на підставі відповідного протоколу можна приймати рішення. Два найвідоміші протоколи, що широко використовуються – це OpenVPN і IPSec, а порівняно недавно з'явився ще WireGuard. Є й інші альтернативи, які вже застарілі, але цілком здатні вирішувати певні завдання.

Вибір конкретного VPN-протоколу визначається низкою обставин і параметрів. На практиці вирішальними можуть бути стабільність і пропускна

здатність інтернет-каналу, сила та реалізація шифрування, яке пропонує провайдер, а також правове поле держави, де перебуває користувач. У деяких країнах роботу VPN суворо регулюють або взагалі забороняють, тож перед підключенням варто переконатися, що сервіс не порушує місцевих законів. У підсумку доцільно звести ключові плюси й мінуси VPN у стислий огляд — саме це і зроблено в таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки VPN

Переваги VPN	**Недоліки VPN**
Захищена передача даних	Потребує налаштування та обслуговування серверів VPN
Конфіденційність інформації	Можливість обмеження швидкості під час використання VPN
Забезпечує анонімність в мережі	Додаткові витрати на підтримку інфраструктури VPN
Доступ до внутрішньої мережі ззовні	Можливість блокування VPN провайдером або країною
Захищений доступ до віддалених ресурсів	Можливість витоку DNS запитів через VPN

VPN — чи не найпоказовіший приклад того, як криптографічні протоколи працюють в Інтернеті на практиці. У таких мережах шифрування гарантує конфіденційність: перехоплені пакети не зможе прочитати ніхто, окрім адресата. Механізми контролю цілісності додатково переконуються, що дані не було непомітно змінено дорогою. Нарешті, автентифікація — через цифрові сертифікати й ключі — підтверджує особу сторін ще до встановлення тунелю. Разом ці складники формують захищений канал, здатний безпечно передавати інформацію навіть у повністю відкритих мережах.

VPN здебільшого покладається на різні криптопротоколи — IPSec, PPTP, SSTP та інші — аби створити захищений тунель. Тож далі зосередимося на тому, як саме ці протоколи працюють у межах VPN-технології.

Point-to-Point Tunneling Protocol (PPTP) — чи не найдавніший із VPN-протоколів, що все ще залишається у вжитку; свого часу його розробила корпорація Microsoft (див. рис. 2.4).

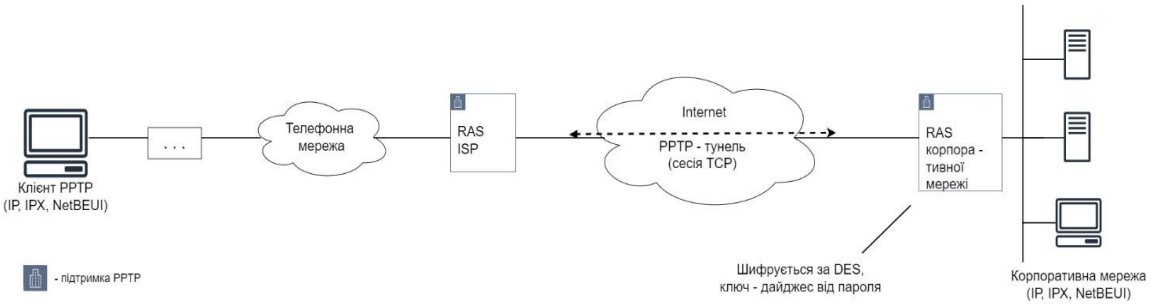

Рисунок 2.4 – Схематичне зображення захищеного каналу PPTP

PPTP працює через два окремі канали: керувальний і даний. Управління ведеться по TCP-сеансі на серверному порту 1723, тоді як корисний трафік інкапсулюється всередині протоколу GRE, який тут виконує роль “транспортного” замінника TCP/UDP. Оскільки GRE не передбачає стандартної процедури встановлення сесії, клієнтам, розміщеним за NAT, зазвичай важко напряму ініціювати point-to-point-зв’язок. Втім розширений варіант GRE, що використовується у PPTP, містить поле Call ID: за ним маршрутизатори можуть зіставити пакети від внутрішнього вузла з відповіддю зовнішнього VPN-сервера. Цей механізм відомий як VPN Pass-Through і підтримується більшістю сучасних домашніх і корпоративних маршрутизаторів, тож навіть клієнти за NAT можуть без проблем користуватись PPTP-тунелем.

PPTP вбудований у всі випуски Windows і сумісний із більшістю інших ОС. Та хоч працює він досить швидко, надійності йому бракує: після розриву сеансу тунель відновлюється повільніше, ніж, скажімо, в OpenVPN.

Сьогодні PPTP вважається застарілим, а сама Microsoft рекомендує переходити на сучасніші VPN-рішення. Якщо вам потрібен тунель лише для обходу блокувань, PPTP ще згодиться, але для справді надійного захисту варто звернутися до безпечніших протоколів.

Secure Socket Tunneling Protocol (SSTP) розроблений компанією Microsoft. Як і PPTP, він не набув широкої популярності у VPN-середовищі, однак серйозних проблем із безпекою, на відміну від PPTP, за ним не помічено. SSTP передає SSL-трафік через стандартний TCP-порт 443, тож легко проходить крізь більшість брандмауерів. Хоча протокол доступний для Linux, RouterOS і SEIL, основна його аудиторія — користувачі Windows-пристроїв.

SSTP залишається доволі легким, оскільки більшість криптографічних операцій перекладає на інші протоколи. По суті, власною у ньому є лише функція cryptographic binding. Шифрування відбувається на рівні SSL, тож увесь трафік — пакети SSTP, PPP та інші — виходить у мережу вже зашифрованим.

Під час встановлення тунелю автентифікація відбувається одразу на трьох рівнях: через SSL, PPP та сам SSTP. Спершу, коли встановлюється SSL-сеанс, клієнт перевіряє сервер за його SSL-сертифікатом. Теоретично сервер може також ідентифікувати клієнта, проте жодна серверна версія Windows такої перевірки не підтримує.

На рівні PPP сервер перевіряє клієнта і, за потреби, сам може пройти перевірку з боку клієнта. Windows Server підтримує такі способи посвідчення: MS-CHAPv2, EAP-TLS, PEAP-MSCHAPv2 і PEAP-TLS. Також доступні PAP (пароль передається відкритим текстом) і CHAP, але ними краще не користуватися, адже вони не обмінюються ключовими даними, потрібними для cryptographic binding. Набір методів збігається з PPTP, однак у SSTP увесь обмін триває вже всередині зашифрованого SSL-каналу.

Internet Protocol Security (IPsec) — це комплект протоколів, що захищає дані під час їхньої мандрівки IP-мережею (див. рисунок 2.5). На відміну від SSL, який працює на прикладному рівні, IPsec діє безпосередньо на мережевому рівні. Його підтримують більшість популярних операційних систем, тож користувачеві не потрібні сторонні програми, як у випадку з OpenVPN.

- Фаза I для вузла A, автентифікація
- Тунель
- Фаза II для вузлів A і B, обмін ключами
- Встановлення тунелю
- Контроль стану тунелю, мінімум кожні 10 с

Рисунок 2.5 - Загальна процедура IPsec

Ключова особливість IPsec у тому, що він шифрує повністю весь IP-пакет: захист застосовується не лише до вмісту, а й до заголовків, які містять службові метадані. Для цього IPsec покладається на два базові механізми. Authentication Header (AH) додає до кожного пакета цифровий підпис, гарантуючи його цілісність і справжність. Encapsulating Security Payload (ESP) забезпечує шифрування, перевірку цілісності та автентифікацію під час передавання трафіку.

IPsec зазвичай комбінується з протоколами IKEv2 (Internet Key Exchange версія 2) та L2TP (Layer 2 Tunneling Protocol) для створення віртуальних приватних мереж (VPN), що гарантують захищену передачу даних. Завдяки цим протоколам забезпечується конфіденційність, цілісність і аутентифікація даних, що передаються через мережу. Розглянемо, як IPsec взаємодіє з L2TP та IKEv2 для створення такого захищеного каналу.

Протокол тунелювання другого рівня (L2TP) був розроблений у 1999 році як вдосконалення протоколів L2F від Cisco та PPTP від Microsoft. Оскільки L2TP сам по собі не забезпечує шифрування та автентифікацію, він часто використовується в парі з IPsec. Це поєднання протоколів, яке отримало назву L2TP/IPsec, підтримується багатьма операційними системами та стандартизоване в RFC 3193.

L2TP/IPsec вважається безпечним і не має суттєвих виявлених вразливостей, що робить його значно більш надійним, ніж PPTP. Протокол може використовувати шифрування 3DES або AES, хоча 3DES, який вже визнаний слабким, застосовується вкрай рідко. L2TP/IPsec здатен забезпечити високий рівень безпеки при передачі даних, простоту налаштування і сумісний з усіма сучасними операційними системами. Проте через подвійну інкапсуляцію даних, L2TP/IPsec може бути менш ефективним і повільнішим порівняно з іншими VPN-протоколами. Крім того, іноді виникають проблеми з використанням L2TP, оскільки він за замовчуванням використовує UDP-порт 500, який можуть блокувати деякі файрволи.

Internet Key Exchange версія 2 (IKEv2) є протоколом для IPsec, який використовується для взаємної аутентифікації, а також для створення та підтримки Security Associations (SA). Протокол стандартизований в RFC 7296. Як і L2TP, IKEv2 використовує IPsec для захисту даних, що свідчить про схожий рівень безпеки між цими протоколами. Хоча IKEv2 був розроблений Microsoft у співпраці з Cisco, існують відкриті реалізації цього протоколу, такі як OpenIKEv2, Openswan та strongSwan.

Завдяки підтримці протоколу Mobility and Multi-homing Protocol (MOBIKE), IKEv2 демонструє високу стійкість до зміни мереж. Це робить його ідеальним вибором для користувачів смартфонів, які часто перемикаються між домашнім Wi-Fi та мобільним з'єднанням або переміщуються між різними точками доступу.

У зв’язці IKEv2 з IPsec можна обрати кілька шифрів — серед них AES, Blowfish і Camellia, причому доступні конфігурації з ключами до 256 біт.

IKEv2 забезпечує режим Perfect Forward Secrecy. Нерідко він дає вищу швидкість, ніж OpenVPN, оскільки потребує менше системних ресурсів. Завдяки швидкому встановленню сесій цей протокол особливо зручний на мобільних пристроях. Підтримку IKEv2 мають Windows (починаючи з версії 7), macOS від 10.11, iOS і частина Android-девайсів.

OpenVPN — універсальний VPN-протокол з відкритим вихідним кодом від OpenVPN Technologies. Нині він вважається одним із найпоширеніших VPN-рішень і, будучи відкритим стандартом, неодноразово проходив незалежні аудити безпеки (див. рисунок 2.6).

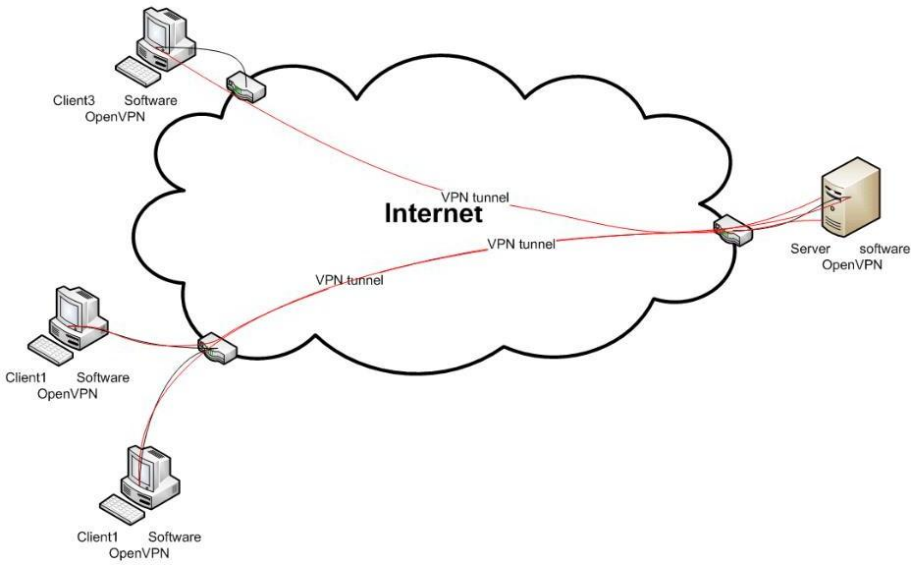

Рисунок 2.6 – Схематичне зображення застосування OpenVPN

У більшості випадків для організації VPN-з'єднання зазвичай обирають OpenVPN. Він відрізняється стабільністю та забезпечує високу швидкість передачі даних. Завдяки підтримці як TCP, так і UDP — стандартних транспортних протоколів — OpenVPN стає зручною альтернативою IPsec, коли провайдер блокує інші VPN-рішення.

Щоб користуватися OpenVPN, потрібен окремий клієнт. Більшість VPN-провайдерів пропонують власні програми на його основі, сумісні з різними операційними системами та пристроями. OpenVPN може працювати через будь-які TCP- або UDP-порти і підтримується на всіх ключових платформах: Windows, macOS, Linux, iOS і Android.

WireGuard — найсвіжіший протокол VPN, який розробники вважають гідною заміною IPsec та OpenVPN у більшості випадків завдяки підвищеній безпеці, продуктивності та простоті налаштування.

Усі IP-пакети, що надходять на інтерфейс WireGuard, упаковуються в UDP і передаються з гарантією безпеки. Для шифрування використовується ChaCha20, обмін ключами здійснюється за допомогою Curve25519, SipHash захищає ключі хеш-таблиць, Poly1305 відповідає за автентифікацію даних, а BLAKE2 застосовується для хешування.

Оскільки реалізація WireGuard містить близько 4 000 рядків коду замість сотень тисяч, її легше перевірити на вразливості, а також швидше розгорнути та налаштувати.

Всю інформацію про швидкодію можна переглянути на офіційному сайті. WireGuard особливо ефективний на Linux-системах, оскільки працює як модуль ядра. Код, включений у ядро, пройшов незалежний аудит безпеки й жодних вразливостей не виявлено. Це обнадійливо, але те, чи зможе WireGuard повністю витіснити IPsec та OpenVPN, покаже час і майбутні незалежні дослідження.

Після розгляду найпоширеніших VPN-протоколів пропоную таблицю 2.2, в якій зібрано їхні ключові технічні характеристики.

Таблиця 2.2 – Важливі технічні характеристики VPN протоколів

Назва	Розробник	Ліцензія	Розгортання	Шифрування	Порти	Недоліки безпеки
PPTP	Microsoft	Proprietary	Windows, macOS, IOS, якийсь час GNU/Linux. Працює "з коробки", не вимагаючи встановлення додаткового ПЗ	Використовує Microsoft Point-to-Point Encryption (MPPE), що реалізує RSA RC4 з максимум 128-бітовими сеансовими ключами	TCP-порт 1723	Має серйозні вразливості. MSCHAP-v2 вразливий для атаки за словником, алгоритм RC4 піддається атаці Bit-flipping.
SSTP	Microsoft	Proprietary	Windows. Працює "з коробки", не вимагаючи встановлення додаткового ПЗ	SSL (шифруються всі частини, крім TCP- та SSL-заголовків)	TCP-порт 443	Серйозних недоліків безпеки не було виявлено.
L2TP/IPsec	12TP - спільна розробка Cisco та Microsoft	Proprietary	Windows, Mac OS X, Linux, IOS. Android. Багато ОС (включно з Windows 2000/XP + Mac OS 10.3+) мають вбудовану підтримку, немає необхідності в додатковому ПЗ	3DES або AES	UDP-порт 500 для початкового обміну ключами та UDP-порт 1701 для початкової конфігурації L2TP, UDP-порт 5500 для обходу NAT	Не вдалося знайти інформації про наявні недоліки безпеки, крім інциденту з витоком доповідей АНБ щодо IPsec.
OpenVPN	OpenVPN Technologies	GNU GPL	Windows, macOS, GNU/Linux Apple IOS, Android та маршрутизатори. Необхідна установка спеціалізованого ПЗ, що підтримує роботу з цим протоколом	Використовує бібліотеку OpenSS (реалізує більшість популярних криптографічних стандартів)	Будь-який UDP- або TCP-порт	Серйозних недоліків безпеки не було виявлено.
WireGuard	Jason A. Donenfeld	GNU GPL	Windows, macOS, GNU/Linux Apple IOS, Android. Встановити сам WireGuard, потім налаштувати згідно інструкцій	Обмін ключами по 1-RTT, Curve25519 для ECDH RFC7539 для ChaCha20 і Poly1305 для автентифікаційного шифрування, і BLAKE2s для хешування	Будь-який UDP-порт	Серйозних недоліків безпеки не було виявлено.

2.4 Порівняльний аналіз криптографічних протоколів захисту інформації в мережі Інтернет та їх версій

2.4.1 Порівняльний аналіз криптографічних протоколів захисту інформації в мережі Інтернет

Кожен із протоколів має свої особливі властивості для захисту даних у мережі. Головна задача криптографічних схем — гарантувати конфіденційність, цілісність і автентичність інформації, а також організувати безпечні канали зв’язку й перевірку користувачів. Кожен протокол поєднує власні алгоритми шифрування, методи обміну ключами та механізми захисту, що визначає його продуктивність, стійкість, ресурсомісткість і зручність у застосуванні.

У цьому розділі розглянемо ключові криптопротоколи — SSL/TLS, IPsec, SSH, OpenPGP, S/MIME, Kerberos, IKEv2 та PGP — і звернемо увагу на їхні відмінні риси та сфери застосування при виборі підходящого рішення.

За підсумками порівняння видно, що SSL/TLS займає лідируючі позиції завдяки широкому розповсюдженню, високій швидкості та стійкості шифрування, простоті налаштування та підтримці різноманітних даних і пристроїв, але його рівень анонімності залишається низьким. IPsec вирізняється надзвичайною стійкістю та повним захистом IP-трафіку на мережевому рівні, однак висока ресурсомісткість і складність розгортання можуть створювати незручності. SSH спроектований спеціально для безпечного віддаленого доступу та передачі даних: він демонструє добрі показники шифрування й стійкості, проте його комфорт використання оцінюється як середній. Для електронної пошти OpenPGP і S/MIME забезпечують надійне шифрування та захист цілісності, але вимагають додаткових зусиль з боку користувача при встановленні та керуванні ключами. Kerberos, призначений для аутентифікації й авторизації в розподілених системах, поєднує середню ефективність шифрування з високою стійкістю та забезпечує достатній рівень анонімності, хоча його впровадження іноді

ускладнює обмежена підтримка. Нарешті, IKEv2 як механізм обміну ключами для створення VPN-тунелів відзначається високою ефективністю шифрування, стійкістю та ресурсомісткістю, а також гнучкістю у виборі алгоритмів шифрування й аутентифікації.

Таблиця 2.3 - Порівняння властивостей криптографічних протоколів в мережі Інтернет

Протокол	**Ефективність шифрування**	**Стійкість**	**Ресурсомісткість**	**Зручність використання та управління**	**Підтримка перевірки цілісності даних**	**Ступінь анонімності**	**Протоколи в основі**
SSL/TLS	Висока	Висока	Середня	Зручна	Так	Низький	RSA, AES, ECC
IPsec	Висока	Висока	Висока	Складна	Так	Високий	DES, 3DES, AES, SHA
SSH	Висока	Висока	Середня	Зручна	Так	Середній	RSA, DSA, AES
OpenPGP	Висока	Висока	Середня	Зручна	Так	Середній	RSA, DSA, AES
S/MIME	Висока	Висока	Середня	Зручна	Так	Середній	RSA, AES
Kerberos	Середня	Висока	Висока	Складна	Так	Високий	DES, RC4, AES
IKEv2	Висока	Висока	Висока	Складна	Так	Високий	AES, SHA
PGP	Висока	Висока	Середня	Зручна	Так	Середній	RSA, DSA, AES

Далі подано таблицю 2.4, яка порівнює ключові характеристики інтернет-криптопротоколів і демонструє, які з них гарантують конфіденційність, цілісність та автентичність даних, а також підтримують створення захищених каналів комунікації й автентифікацію користувачів.

Таблиця 2.4 – Порівняння ключових властивостей криптографічних протоколів в мережі Інтернет

Протокол	**Забезпечення конфіденційності**	**Забезпечення цілісності**	**Забезпечення аутентичності**	**Забезпечення безпечних каналів**	**Аутентифікація користувачів**
SSL/TLS	+	+	+	+	+
IPsec	+	+	+	+	+
SSH	+	+	+	+	+
OpenPGP	+	+	+	-	-
S/MIME	+	+	+	-	-
Kerberos	-	-	+	-	+
IKEv2	+	+	+	+	+
PGP	+	+	+	-	-

За результатами аналізу таблиці 2.4 можна зробити висновок, що SSL/TLS забезпечує високий рівень конфіденційності, цілісності й автентичності даних завдяки шифруванню, перевірці цілісності повідомлень і взаємній автентифікації клієнта та сервера, при цьому протокол підтримує створення захищених каналів зв’язку й різні механізми підтвердження особи. IPsec, як набір мережевих протоколів, шифрує всі IP-пакети та перевіряє їхню цілісність, що дозволяє організовувати безпечні тунелі й автентифікувати учасників обміну даними на рівні IP. Secure Shell (SSH) призначений для віддаленого захищеного доступу й передачі даних: він поєднує потужне шифрування з надійними методами аутентифікації і може використовуватися для побудови тунелів.

Протоколи OpenPGP і S/MIME служать для шифрування та цифрового підпису електронної пошти й файлів, обидва гарантують конфіденційність і

цілісність, але не передбачають організації власних захищених каналів або вбудованої автентифікації користувачів. Kerberos, навпаки, зосереджений на централізованій автентифікації та авторизації в розподілених системах: він забезпечує перевірку особи й керування правами користувачів, проте не відповідає за шифрування вмісту передаваних повідомлень.

IKEv2 виконує роль протоколу обміну ключами для VPN-з'єднань, поєднуючи конфіденційність, цілісність та автентичність даних із підтримкою захищених каналів і різних методів аутентифікації. Нарешті, Pretty Good Privacy (PGP) — це рішення для шифрування, підпису й захисту електронних повідомлень і файлів, яке, як і OpenPGP, не включає власних засобів організації безпечного транспортного каналу чи автентифікації учасників.

2.4.2 Порівняльний аналіз версій криптографічних протоколів захисту інформації в мережі Інтернет

Коли ми розглядаємо інтернет-криптопротоколи, важливо оцінювати їхню здатність гарантувати конфіденційність, цілісність, автентичність і доступність даних. Протоколи типу SSL/TLS, SSH чи IPsec забезпечують різні рівні захисту в залежності від обраних алгоритмів шифрування, методів автентифікації та способів обміну ключами. Перед тим як заглиблюватися в особливості конкретного протоколу, варто дослідити його історію, порівняти різні версії та проаналізувати відомі вразливості. У таблицях 2.5 і 2.6 наведено приклад такого аналізу для протоколу TLS — одного з найпоширеніших в Інтернеті — із фокусом на версіях TLS 1.0, 1.1, 1.2, 1.3 та попередніх ревізіях SSL 1.0, 2.0 і 3.0.

Таблиця 2.5 – Порівняння версій SSL 1.0, 2.0 та 3.0

Версія	Рік стандартизації	Оновлення / поліпшення	Виявлені проблеми безпеки та вразливості	Інші характеристики
SSL 1.0	1995	Перша версія протоколу, яка була розроблена компанією Netscape. Ніколи не була опублікована через виявлені в ній серйозні вразливості	Містить багато вразливостей, через що ніколи не була офіційно випущена	Ця версія ніколи не була загальнодоступна
SSL 2.0	1995	Усунення деяких вразливостей, виявлених у SSL 1.0. Підтримка механізмів аутентифікації та захисту передачі даних	Містить вразливості до атак на повторення сесій та проблеми з автентифікацією сервера	Зараз вважається застарілим і не безпечним
SSL 3.0	1996	Більш сучасні алгоритми шифрування, виправлення вразливостей SSL 2.0.	Став жертвою атаки POODLE (Padding Oracle On Downgraded Legacy Encryption), яка дозволила зловмисникам розшифровувати зашифровану інформацію	Зараз вважається застарілим і не безпечним, замінений на TLS 1.0

SSL 1.0 так і не потрапив у масовий обіг через серйозні дірки в безпеці, які були виявлені ще на ранніх стадіях розробки. Невдовзі після цього виявили слабкі сторони і в SSL 2.0, тож він залишився здебільшого експериментальним і здобув погану репутацію. SSL 3.0 протримався довше, але з появою атаки POODLE стало очевидно, що його модель шифрування не витримує сучасних методів криптоаналізу й дозволяє обхід захисту навіть за звичайного веб-трафіку. Саме тому розробники браузерів і серверного ПЗ почали відключати підтримку цих протоколів “за замовчуванням”: сучасні релізи Apache, Nginx і навіть Windows Server більше не допускають SSLv2/v3 без явної ручної активації.

Сьогодні застарілі версії SSL розглядають лише в контексті історичних досліджень і сумнівних сценаріїв сумісності зі старими пристроями. Реальні ж системи вимагають мінімум TLS 1.2 або, краще за все, TLS 1.3 — не лише заради уникнення відомих атак, а й для підтримки більш ефективних шифрів і

прискорених алгоритмів обміну ключами. Якщо ж із якоїсь причини вам усе ще необхідно працювати з давніми клієнтами, варто окремо зважити ризики та, можливо, організувати проксі-шар, який би безпечно конвертував трафік між застарілими та сучасними стандартами.

Таблиця 2.6 – Порівняння версій TLS 1.0, 1.1, 1.2 та 1.3

Версія	Рік стандартизації	Оновлення / поліпшення	Виявлені проблеми безпеки та вразливості	Інші характеристики
TLS 1.0	1999	Покращений варіант SSLv3 з підтримкою більш сучасних алгоритмів шифрування та механізмом шифрування	Вразливості до атак типу man-in-the-middle (POODLE), а також проблеми з блоковим шифруванням (BEAST)	Часто вважається застарілим та не безпечним
TLS 1.1	2006	Захист від атаки CBC replay, яка використовувала IV; підтримка шифрування, заснованого на AEAD	Багато вразливостей, що могли дозволити атаки посередника (man- in-the-middle)	Вже вважається застарілим та не безпечним
TLS 1.2	2008	Підтримка більш сучасних алгоритмів шифрування; покращена гнучкість при виборі алгоритмів	Чутливість до атак на Padding Oracle (POODLE) та атаки на виробничий процес рандомізації (Lucky 13)	Надає більш безпечний та гнучкий вибір шифрування
TLS 1.3	2018	Скорочення часу рукостискання; підтримка Forward Secrecy; усунення старих і небезпечних алгоритмів шифрування	Немає широко відомих вразливостей на момент написання	Набагато швидше рукостискання, ніж у попередніх версіях; зміцнена безпека

Перехід із SSL на TLS відбувся невипадково—його метою було підвищення рівня безпеки передавання даних. Хоча згодом і TLS 1.0, і 1.1, і 1.2 поступово втрачали надійність через виявлені вразливості та вдосконалені

методи атак, версія TLS 1.3 демонструє значно кращий захист і наразі є актуальною, допоки не з’явиться TLS 1.4.

Як і у випадку з TLS, розвиток IPsec супроводжується регулярними оновленнями для усунення вразливостей та підвищення рівня безпеки. Протокол обміну ключами IKE — ключовий елемент IPsec — теж зазнав еволюції: спочатку з’явився IKEv1, потім його замінив IKEv2, у якому враховані недоліки попередника й реалізовані нові механізми захисту. У таблиці 2.7 ми порівняємо ці дві версії IKE, звернувши увагу на їхні відмінності в способах обміну ключами й управлінні безпековими асоціаціями.

Таблиця 2.7 – Порівняння версій компонента IPsec - IKE

Версія	**Рік стандартизації**	**Оновлення / поліпшення**	**Виявлені проблеми безпеки та вразливості**	**Інші характеристики**
IKEv1	1998	Перший стандарт для IPsec key management	Aggressive Mode, numerous RFC complaints	Використовується в багатьох старіших системах, не рекомендується для нових розробок через вразливості
IKEv2	2005	Більш стабільний і безпечний, зменшена складність	Fewer known vulnerabilities	Використовується в сучасних системах заради його кращої безпеки і ефективності, більші можливості розширення

Регулярна актуалізація криптопротоколів — це не просто формальність, а необхідний крок для підтримки безпечної роботи будь-якої мережевої інфраструктури. Під час кожного випуску розробники виправляють знайдені вразливості, оптимізують алгоритми шифрування та враховують нові методи атак, які з’явилися в інтернеті. Якщо залишатися на старих версіях, система стає відкритою для перехоплення трафіку, несанкціонованого доступу до

даних або підміни інформації – а це вже не просто технічний ризик, а безпосередня загроза репутації компанії й безпеці користувачів.

До того ж, впровадження оновлень слід проводити з урахуванням сумісності з існуючою інфраструктурою: перед масовим розгортанням корисно протестувати нові релізи в окремому середовищі, щоб переконатися, що вони не порушують роботу додатків чи не створюють конфліктів. Після цього можна планомірно поширювати оновлені налаштування на основні сервери, супроводжуючи процес моніторингом і логуванням, щоб у разі непередбачених ситуацій одразу виявляти й усувати можливі неполадки.

Для підтримки високого рівня безпеки важливо також зв’язати управління протоколами з внутрішніми політиками безпеки: документувати терміни повторної перевірки, формалізувати процедуру реагування на нові вразливості й навчати команду адміністраторів.

РОЗДІЛ 3 ДОСЛІДЖЕННЯ БЕЗПЕКИ ВЕБСЕРВЕРА ДЛЯ РІЗНИХ ВЕРСІЙ ПРОТОКОЛІВ SSL ТА TLS

3.1 Оцінка стану безпеки вебсервера

3.1.1 Опис об’єкту дослідження

У межах цього дослідження аналізується вебсервер, що функціонує під управлінням операційної системи FreeBSD.

FreeBSD — відкрита операційна система на базі BSD, успадкована від Unix; її відзначають потужні мережеві можливості, висока надійність і стабільність.

FreeBSD вирізняється видатною продуктивністю та стабільністю, через що її часто обирають для великих серверних інсталяцій у розподілених мережах. Гнучкість налаштувань дозволяє адміністраторам тонко конфігурувати будь-які компоненти системи — від параметрів мережі й розподілу ресурсів до служб і застосунків — під конкретні вимоги інфраструктури. Окрім цього, FreeBSD постачається з широким набором інструментів безпеки: вбудованими механізмами ізоляції процесів, мережевого фільтрування, аудиту та шифрування, що дає змогу створити надійні бар’єри проти несанкціонованого доступу і атак.

FreeBSD відзначається високою масштабованістю: її можна сконфігурувати для обробки інтенсивних навантажень із застосуванням різних технологій розподілу та балансування запитів. Завдяки цьому система здатна одночасно обслуговувати велику кількість клієнтських запитів, зберігаючи при цьому стабільність і надійність роботи.

FreeBSD пропонує розвинену екосистему для вебсерверів: вона сумісна з такими популярними рішеннями, як Apache, Nginx, Lighttpd та іншими, а також підтримує встановлення численних модулів і плагінів для розширення функціональності. Активна спільнота розробників і регулярні оновлення безпеки, спрямовані на виявлення й усунення вразливостей, забезпечують

платформі високу стабільність і надійність, що робить FreeBSD одним із найпопулярніших виборів для розгортання вебсерверних середовищ.

Об’єкт дослідження – вебсервер на базі Apache. Цей сервер входить до числа найпоширеніших у світі завдяки своїй надійності, гнучкості та здатності масштабуватися під різні завдання. Завдяки відкритому коду кожен бажаючий може вивчити його внутрішню структуру, внести зміни чи покращення, що стимулює постійний розвиток і вдосконалення проекту спільнотою розробників у всьому світі.

Однією з ключових особливостей Apache є можливість підключати численні модулі для різноманітних завдань, зокрема для шифрування трафіку. За допомогою таких розширень сервер отримує здатність застосовувати SSL/TLS, що гарантує захищену передачу даних між клієнтом і сервером. Це стає вкрай важливим при роботі з конфіденційною інформацією — паролями, персональними даними чи фінансовими транзакціями — оскільки запобігає несанкціонованому доступу до критичних відомостей.

Вебсервер Apache на FreeBSD приймає з’єднання через стандартні порти: 80 використовується для HTTP, а 443 — для HTTPS. Перший є звичайним протоколом передачі даних у Мережі, тоді як другий — його захищена версія з шифруванням на основі SSL/TLS.

Стандартне призначення портів сприяє надійному захисту даних в Інтернеті. Зокрема, коли HTTPS працює на порті 443 і використовує шифрування SSL/TLS, це запобігає прослуховуванню та несанкціонованому доступу до інформації. Такий підхід забезпечує конфіденційність і цілісність обміну між вебсервером і клієнтом, що є критично важливим для безпечної взаємодії з веб-додатками.

3.1.2 Устаткування поточного стану безпеки вебсервера та аналіз виявлених вразливостей

У процесі оцінки безпеки вебсервера застосовували сканер вразливостей Nessus, щоб виявити потенційні ризики та слабкі місця. За його допомогою виконали перевірку програмного забезпечення, пошук несерйозних паролів, аналіз конфігурацій безпеки та інших факторів, які можуть становити загрозу системі. Nessus, будучи спеціалізованим інструментом, дозволив отримати детальний звіт про знайдені вразливості та оцінити рівень ризику для сервера.

Після налаштування параметрів сканування запускають Nessus для перевірки вебсервера. Інструмент обходить відкриті порти та аналізує запущені сервіси на наявність відомих вразливостей. Додатково він виконує сканування використовуваних мережевих протоколів.

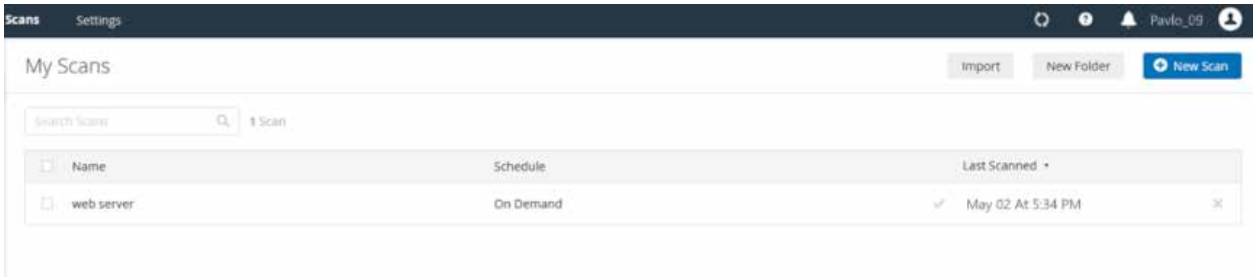

Рисунок 3.1 – Вікно My Scans

Після завершення сканування Nessus формується детальний звіт із підсумками аналізу (див. рис. 3.2). У ньому наведено виявлені вразливості, оцінку їхньої критичності та рекомендації з усунення.

У звіті також подано розгорнуту інформацію про кожну вразливість: її опис, потенційні наслідки та рекомендації щодо захисних заходів.

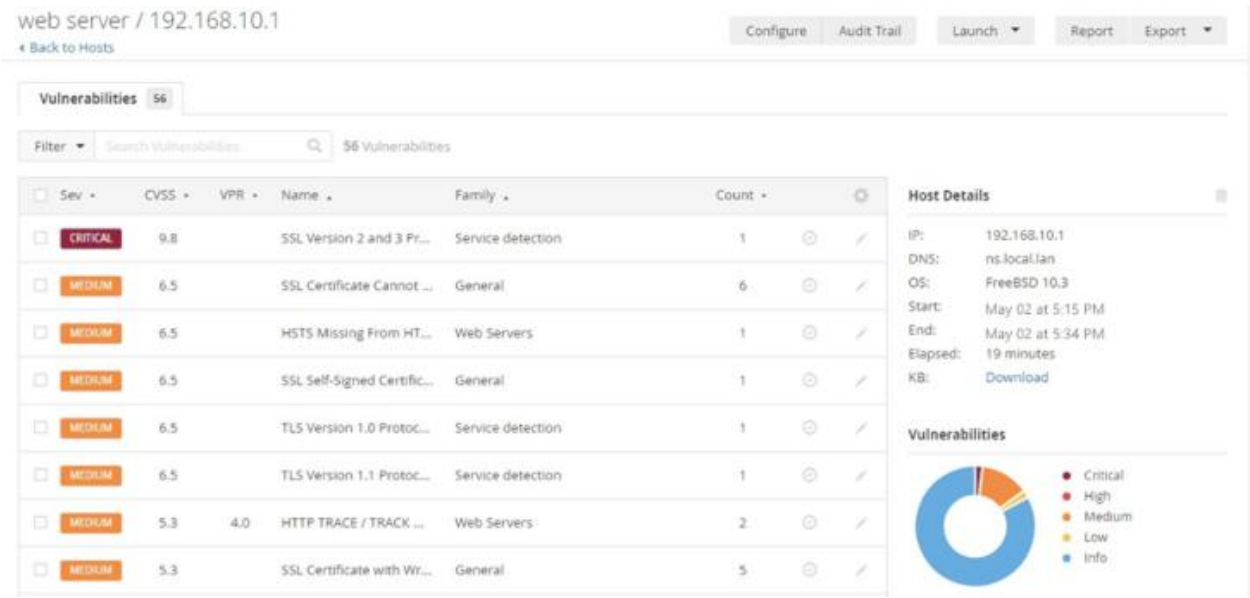

Рисунок 3.2 – Результат Nessus сканування

Аналіз результатів сканування показав, що на сервері активовано застарілі й небезпечні протоколи SSL 2 та SSL 3 (ідентифікатор «SSL version 2 and 3 Protocol Detection»), через що виникає критична загроза — зловмисники можуть отримати несанкціонований доступ або здійснити шкідливі дії. Окрім цього, виявлено помірні уразливості, пов’язані з підтримкою TLS 1.0 та TLS 1.1, які також мають відомі прогалини в захисті і створюють додаткові ризики для безпеки системи.

Оскільки виявлені уразливості становлять значну загрозу, слід невідкладно відмовитися від TLS 1.0, TLS 1.1 та всіх версій SSL і натомість увімкнути підтримку сучасніших стандартів — TLS 1.2 або TLS 1.3, залежно від можливостей вашого сервера. Це значно знизить ризик потенційних атак і підвищить загальний рівень безпеки системи.

Перш ніж братися за усунення виявлених уразливостей, варто усвідомити, що не всі з них потребують негайної обробки. Слід визначити пріоритети, щоб раціонально розподілити ресурси й час відповідно до критичності кожної вразливості та її потенційного впливу на систему.

Після ідентифікації та аналізу вразливостей слід перейти до визначення черговості їх усунення. Дані про рівень критичності, які надає сканер, є

відмінною відправною точкою для цього планування. Завдяки цьому можна зрозуміти, які проблеми вимагають першочергового вирішення.

У нашому випадку виявлено одну критичну вразливість — детектування SSL версій 2 і 3. Це свідчить про високий рівень ризику та може спричинити серйозні наслідки для системи у разі експлуатації зловмисниками. Тому цю проблему слід усунути якнайшвидше.

Вразливості середнього рівня — зокрема «SSL Certificate Cannot Be Trusted», «SSL Self-Signed Certificate», виявлення протоколів TLS 1.0 і TLS 1.1 та подібні — також слід усунути. Однак через їхню відносно нижчу критичність їх виправлення можна відкласти до усунення критичної загрози, особливо за обмежених ресурсів.

Визначення порядку виправлень потребує продуманого підходу та зваженого розподілу ресурсів з урахуванням рівня небезпеки кожної вразливості. Такий метод дозволяє сконцентрувати зусилля на найбільш критичних проблемах і гарантує, що найсерйозніші загрози усуваються в першу чергу.

3.2 Аналіз безпеки вебсервера для версій протоколів SSL та TLS

3.2.1 Структура конфігураційного файлу

Після проведеного сканування й оцінки стану безпеки було виявлено низку загроз, які ми проаналізували та розставили за пріоритетністю за рівнем їхнього впливу. Наступним етапом стане розробка стратегії та плану усунення цих уразливостей.

Основний захід полягає в оновленні налаштувань у файлі httpd-ssl.conf. У FreeBSD цей конфігураційний файл, який зазвичай знаходиться за шляхом /usr/local/etc/apache24/extra/httpd-ssl.conf відповідає за параметри SSL/TLS для Apache HTTP Server. Водночас його розташування може відрізнятися залежно від конкретних версій FreeBSD або Apache.

У цьому файлі задаються параметри модуля SSL/TLS для Apache HTTP Server. У Лістингу 1.1 наведено уривок із конфігураційного файлу httpd-ssl.conf із найважливішими директивами для SSL і TLS.

Лістинг 1.1 - Фрагмент вмісту конфігураційного файлу httpd-ssl.conf з ключовими налаштуваннями

```
Listen 443

SSLCipherSuite
HIGH:MEDIUM:!MD5:!RC4:!3DES
SSLProxyCipherSuite
HIGH:MEDIUM:!MD5:!RC4:!3DES

SSLHonorCipherOrder on

SSLProtocol all -SSLv3 -SSLv2 +TLSv1 +TLSv1.1 +TLSv1.2
+TLSv1.3

SSLProxyProtocol all -SSlv3 -SSLv2 -TLSv1.2 -TLSv1.3
SSLPassPhraseDialog builtin

SSLSessionCache "shmcb:/var/run/ssl_scache(512000)"
SSLSessionCacheTimeout 300

<VirtualHost _default_:443>

DocumentRoot
"/usr/local/www/apache24/data"
ServerName www.local.lan:443

ServerAdmin admin@local.lan
ErrorLog "/var/log/httpd-
error.log"

TransferLog "/var/log/httpd-access.log"
SSLEngine on

SSLCertificateFile "/usr/local/etc/apache24/server.crt"
SSLCertificateKeyFile "/usr/local/etc/apache24/server.key"

<FilesMatch
"\.(cgi|shtml|php)$">
SSLOptions +StdEnvVars

</FilesMatch>

<Directory
    "/usr/local/etc/apache24/cgi-bin">
    SSLOptions +StdEnvVars

</Directory>

BrowserMatch "MSIE [2-5]"\
    nokeepalive ssl-unclean-
```

```
        shutdown\ downgrade-1.0
        force-response-1.0
    CustomLog "/var/log/httpd-ssl_request.log"\
        "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
    </VirtualHost>
```

У цьому конфігураційному файлі задаються параметри SSL/TLS для Apache HTTP Server. Перш за все, директива Listen 443 перенаправляє сервер на прослуховування порту 443 для HTTPS, тоді як порт 80 залишається призначеним для HTTP. Налаштування SSLCipherSuite і SSLProxyCipherSuite визначають набір криптографічних алгоритмів, обмежуючи список лише сильними і середнього рівня шифрами та виключаючи MD5, RC4 і 3DES, а опція SSLHonorCipherOrder змушує сервер обирати шифр у порядку їхнього перерахування – від найнадійнішого до менш надійного. За допомогою SSLProtocol і SSLProxyProtocol вмикаються всі версії протоколів, окрім застарілих SSLv2 і SSLv3, і одночасно деактивуються TLS 1.2 і 1.3 для проксі. Розшифровка приватних ключів відбувається через вбудований діалог SSLPassPhraseDialog, а механізм кешування сеансів налаштовано через SSLSessionCache із використанням сховища shmcb і таймаутом у 300 секунд.

У секції VirtualHost default:443 прописано шлях до кореневої папки вебдокументів (DocumentRoot "/usr/local/www/apache24/data") та ім’я віртуального хоста (ServerName www.local.lan:443). Директива SSLEngine on активує SSL для цього хоста, а SSLCertificateFile і SSLCertificateKeyFile вказують шляхи до файлів сертифіката й приватного ключа. Для сумісності зі старими браузерами Internet Explorer (версії 2–5) використано BrowserMatch із опціями nokeepalive, ssl-unclean-shutdown, downgrade-1.0 і force-response-1.0. І нарешті, за допомогою CustomLog "/var/log/httpd-ssl_request.log" "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x "%r" %b" налаштовано детальний запис кожного SSL-запиту з інформацією про час, адресу клієнта, протокол, шифр, сам запит і розмір відповіді.

Завдяки цим і подібним налаштуванням ми можемо тонко відрегулювати параметри SSL/TLS та повною мірою використати їхній захисний потенціал. У наступному розділі виконаємо оптимізацію вебсервера, внісши корективи в розглянуті директиви SSL/TLS.

3.2.2 Усунення виявлених критичних вразливостей вебсервера

Після виконання всіх підготовчих кроків — оцінки безпеки, аналізу виявлених уразливостей, визначення пріоритетів та розробки плану дій — можемо перейти до їхнього усунення в установленому порядку. Почнемо з відключення SSL версій 2 і 3 (SSL version 2 and 3 Protocol Detection). Для цього відредагуємо файл httpd-ssl.conf, який містить налаштування модуля SSL/TLS для Apache HTTP Server. У терміналі під користувачем root за допомогою команди mc відкриваємо консольний файловий менеджер Midnight Commander (див. рис. 3.3).

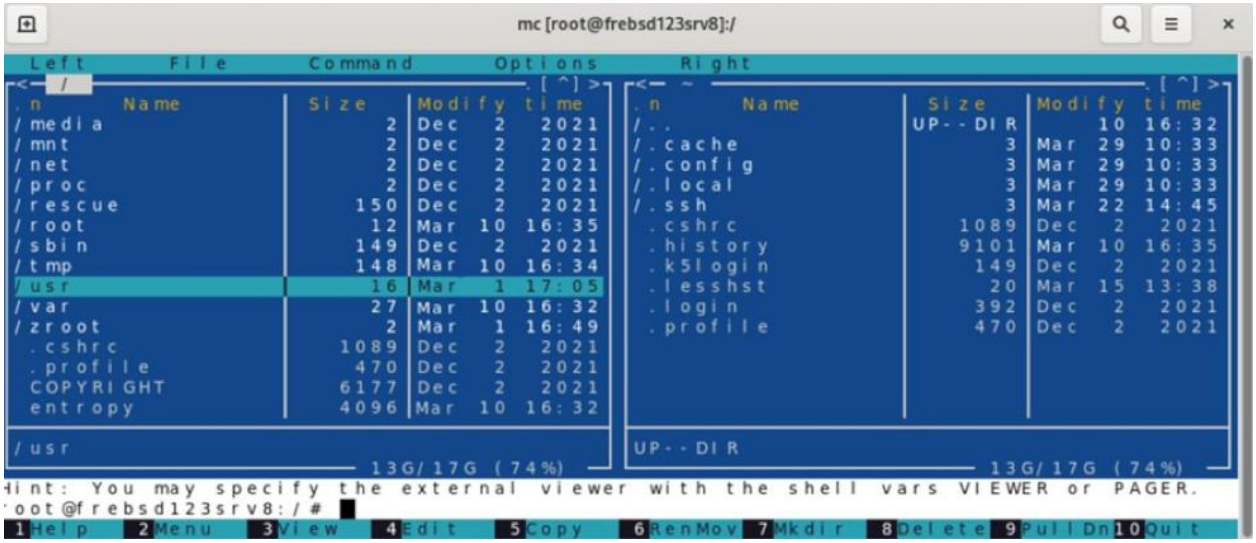

Рисунок 3.3 – Кореневий каталог файлового менеджера Midnight Commander

Заходимо в каталог /usr/local/apache24/extra і відкриваємо для редагування файл httpd-ssl.conf. На ілюстрації 3.3 подано невеликий фрагмент його вмісту. У директиві SSLProtocol all +SSLv3 +SSLv2 +TLSv1 +TLSv1.1 ми

бачимо, що в системі активовані застарілі протоколи SSL 3.0, SSL 2.0, TLS 1.0 та TLS 1.1 (рис. 3.4) — саме ті, які були визнані вразливими під час сканування.

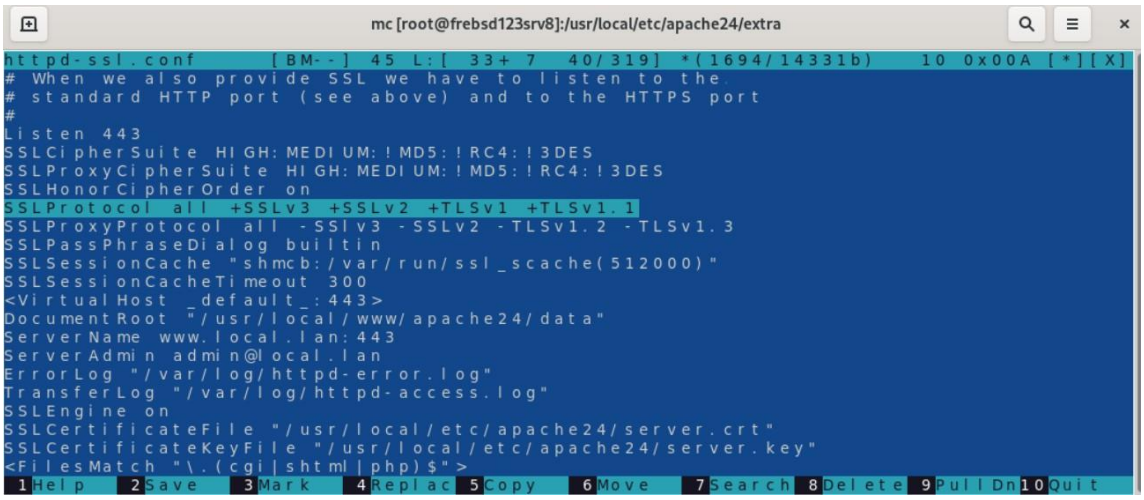

Рисунок 3.4 – Початкові налаштування сертифікатів системи в httpd-ssl.conf

Спершу, згідно з пріоритетами, вимикаємо SSL 3.0 і SSL 2.0 через їхню критичну вразливість. Для цього в конфігураційному файлі замінюємо рядок SSLProtocol all +SSLv3 +SSLv2 +TLSv1 +TLSv1.1 на SSLProtocol all -SSLv3 -SSLv2 +TLSv1 +TLSv1.1 і зберігаємо внесені зміни (див. рис. 3.5).

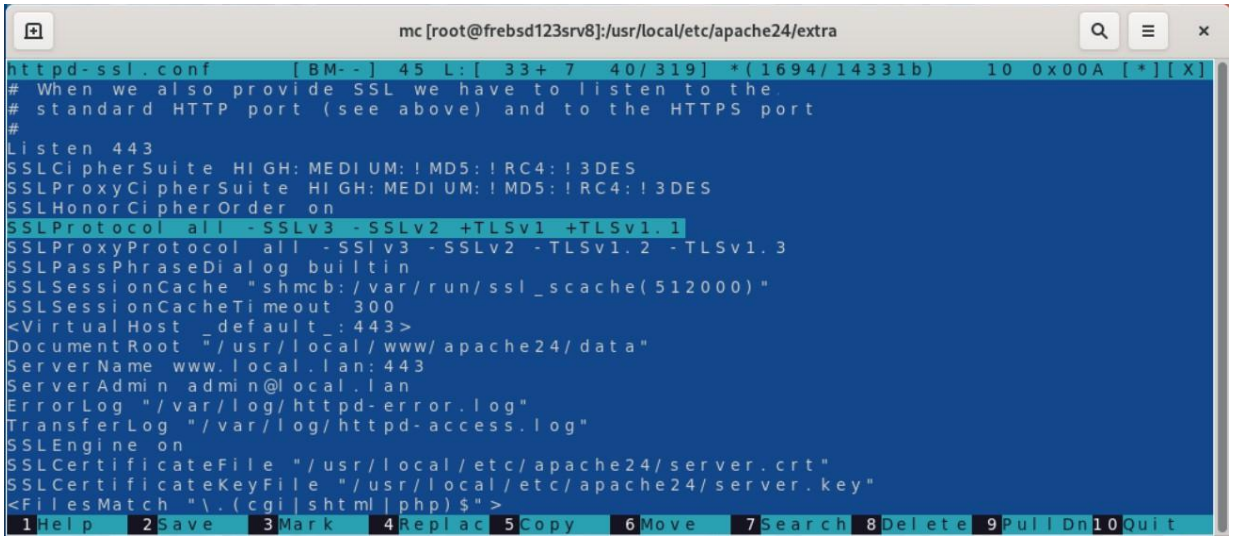

Рисунок 3.5 – Вміст конфігураційного файлу після внесення змін

Далі у терміналі під обліковим записом root виконуємо service apache24 restart для застосування змін у конфігураційних файлах Apache 2.4 без повного

відключення сервера. Ця команда автоматично перевіряє синтаксис файлів, зупиняє поточні процеси Apache і запускає їх заново з оновленими налаштуваннями. На ілюстрації 3.6 показано, як система підтверджує коректність конфігурації, припиняє роботу старих процесів і успішно стартує вебсервер із новими параметрами.

```
root@frebsd123srv8:/usr/local/etc/apache24/extra# service apache24 restart
Performing sanity check on apache24 configuration:
Syntax OK
Stopping apache24.
Waiting for PIDS: 1002.
Performing sanity check on apache24 configuration:
Syntax OK
Starting apache24.
root@frebsd123srv8:/usr/local/etc/apache24/extra#
```

Рисунок 3.6 – Перезапуск сервера Apache

Після рестарту Apache з оновленими налаштуваннями виконаємо ще одну перевірку сканером, щоб упевнитися, що вразливість усунено.

Як показує повторне сканування на рис. 3.7, уразливість «SSL version 2 and 3 Protocol Detection» більше не фіксується, отже її вдалося повністю ліквідувати. Це свідчить про те, що впроваджені заходи ефективні й дозволяють підтримувати стабільну та безпечну роботу сервера.

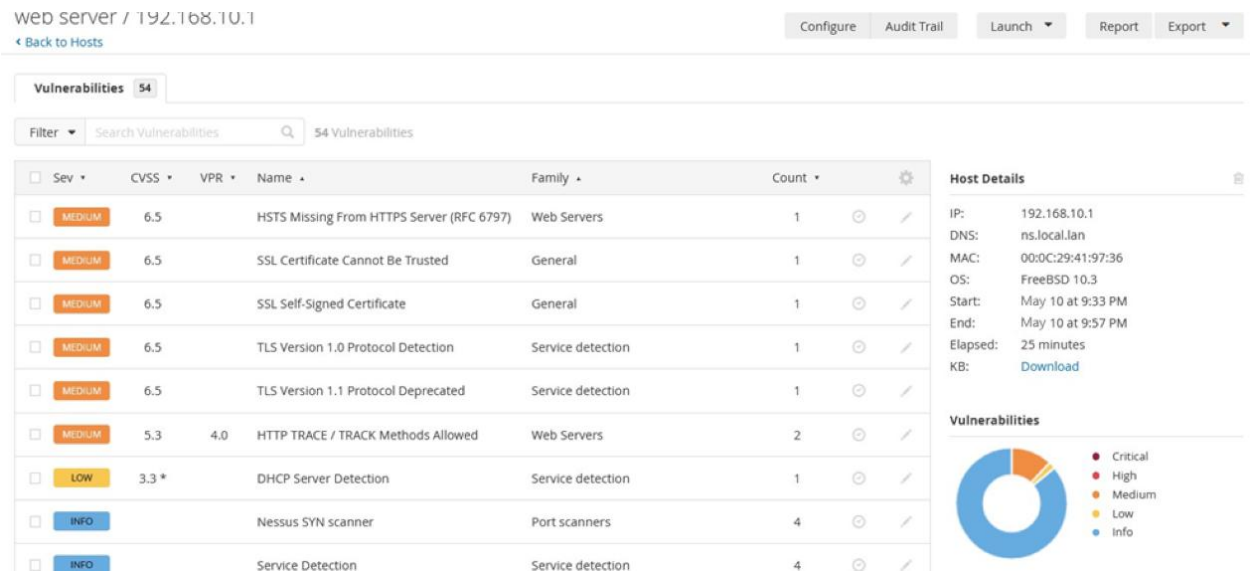

Рисунок 3.7 – Результат повторного сканування

Проте це не кінець роботи. Попри усунення найкритичніших прогалин, у системі ще зберігаються уразливості середнього рівня. Хоч вони й не несуть такої прямої загрози, як критичні, та можуть бути використані зловмисниками для несанкціонованого доступу або шкоди нашій інфраструктурі.

3.2.3 Усунення виявлених середніх вразливостей вебсервера

Після успішного усунення критичних прогалин слід звернути увагу на вразливості середнього рівня. Хоч вони й не завдають такої серйозної шкоди, залишати їх без реакції було б помилкою. До цієї категорії потрапляють недоліки в конфігураціях та налаштуваннях, проблеми сумісності й використання застарілих версій програмного забезпечення.

З метою усунення вразливостей, пов’язаних із застарілими версіями TLS, спочатку відкриваємо файл /usr/local/apache24/extra/httpd-ssl.conf, у якому задаються параметри модуля SSL/TLS для Apache. У його поточному вмісті директива SSLProtocol виглядає так: SSLProtocol all -SSLv3 -SSLv2 +TLSv1 +TLSv1.1. Це означає, що сервер вже відключив небезпечні SSLv2 та SSLv3, проте продовжує підтримувати протоколи TLS 1.0 і TLS 1.1, які також мають відомі прогалини в безпеці. Щоб повністю позбутися цих середніх за рівнем ризику реалізацій, необхідно змінити рядок на: SSLProtocol all -SSLv3 -SSLv2 -TLSv1 -TLSv1.1 +TLSv1.2 +TLSv1.3. Таким чином ми забороняємо підтримку всіх застарілих версій (SSL 2.0, SSL 3.0, TLS 1.0 і TLS 1.1) і водночас активуємо лише сучасні й безпечні протоколи TLS 1.2 та TLS 1.3.

Після здійснених змін, вебсервер Apache буде підтримувати тільки протоколи TLS версій 1.2 та 1.3, а протоколи TLS 1.0 та 1.1 будуть відключені, це має виправити такі вразливості, як TLS version 1.0 Protocol Detection та TLS version 1.0 Protocol Deprecated.

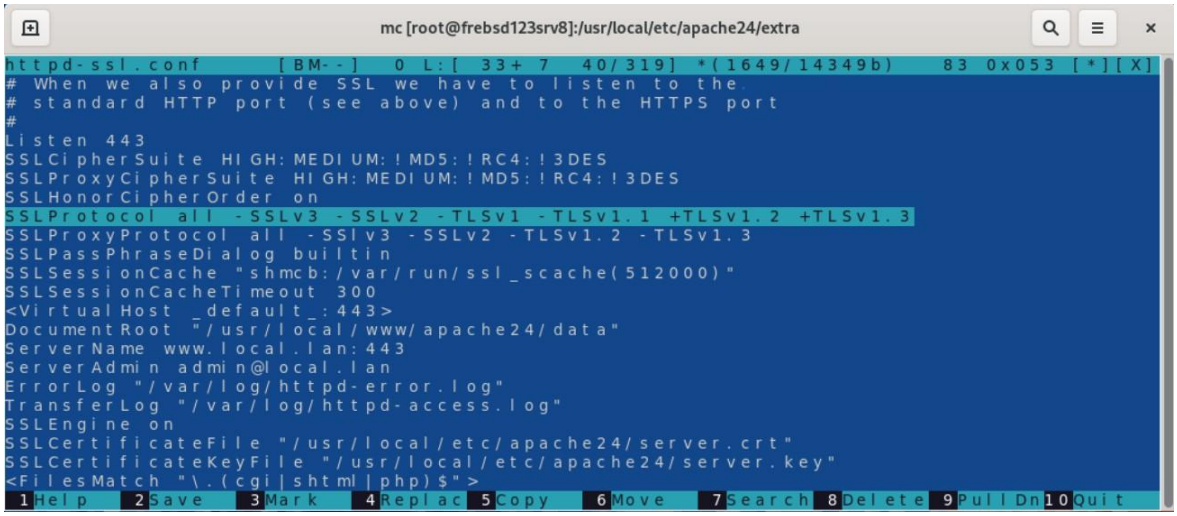

Рисунок 3.8 – Безпечні налаштування версій протоколів вебсерверу Apache

Після збереження змін у файлі налаштувань, так само, як і раніше, виконуємо # service apache24 restart під правами root для перезапуску Apache. На рисунку 3.9 видно, що перевірка синтаксису минула успішно і нові параметри вступили в дію.

```
root@frebsd123srv8:/usr/local/etc/apache24/extra# service apache24 restart
Performing sanity check on apache24 configuration:
Syntax OK
Stopping apache24.
Waiting for PIDS: 1041.
Performing sanity check on apache24 configuration:
Syntax OK
Starting apache24.
root@frebsd123srv8:/usr/local/etc/apache24/extra#
```

Рисунок 3.9 – Повторний перезапуск серверу Apache

Після оновлення конфігураційного файлу для усунення вразливостей середнього рівня виконаємо повторне сканування за допомогою Nessus, щоб пересвідчитися в тому, що всі проблеми були успішно ліквідовані.

На рисунку 3.10 видно, що Nessus більше не фіксує жодних проблем, пов’язаних з TLS 1.0 (Protocol Detection та Protocol Deprecated). Отже, правки в конфігураційному файлі httpd-ssl.conf успішно зміцнили безпеку сервера.

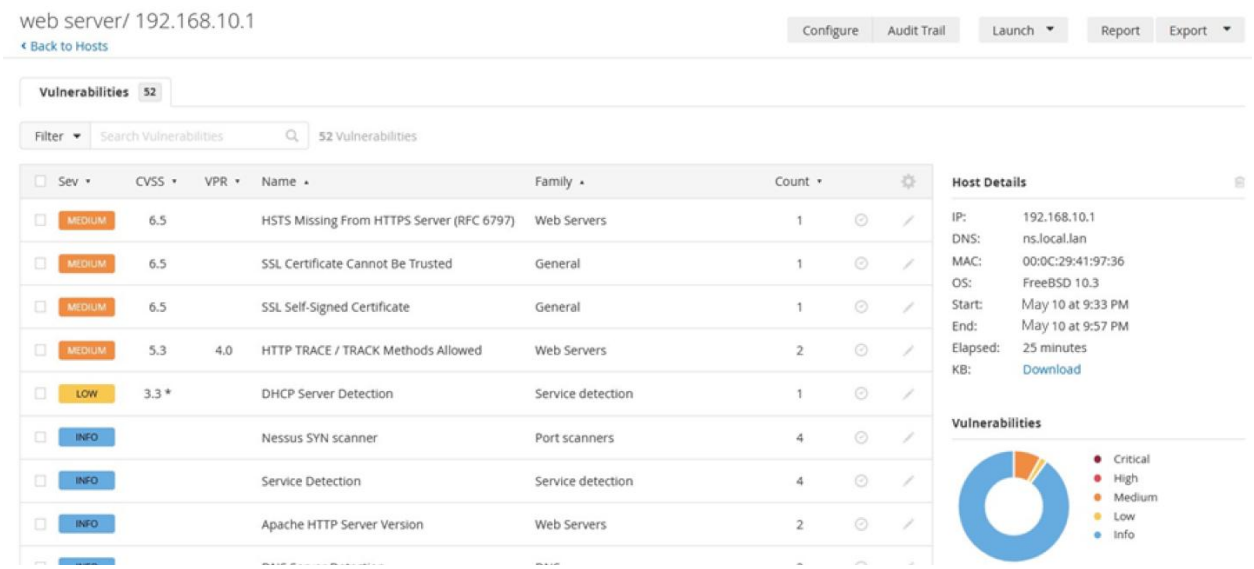

Рисунок 3.10 – Результат сканування після виправлення вразливостей

На рисунку 3.10 видно, що сервер і надалі працює з сертифікатом, якому не вдається здобути довіру в браузерів та систем безпеки, а також із самопідписаним SSL-сертифікатом. У першому випадку йдеться про ситуацію, коли виданий веб-сайтом сертифікат не визнано жодним довіреним центром сертифікації або його термін дії минув, тож браузери попереджають про небезпечність з'єднання. У другому випадку замість сертифіката, підписаного авторитетною організацією, використовується власноруч згенерований ключ, який не має цифрового підтвердження з боку визнаної сертифікаційної служби, через що неможливо автоматично перевірити його автентичність.

Щоб позбавитися помилок "SSL Certificate Cannot Be Trusted" і "SSL Self-Signed Certificate", потрібно встановити сертифікат від авторитетного центру сертифікації. Проте в рамках наших тестів і досліджень головна увага приділяється саме перевіркам роботи сервера, а не адмініструванню сертифікатів. Оскільки самопідписані ключі не впливають на точність експериментів і не підривають безпеку тестового середовища, на цьому етапі можна відкласти їхню заміну. Водночас слід зауважити, що для публічних або виробничих серверів обов'язково слід використовувати дійсні сертифікати, видані визнаними центрами, аби гарантувати довіру й безпечність обміну для кінцевих користувачів.

При перевірці системи Nessus позначив проблему «HSTS Missing from HTTPS Server (RFC 6797)», проте вона виявилася хибнопозитивною і на практиці не потребує виправлення. Подібні помилки виникають, коли сканер невірно інтерпретує відсутність певного заголовка або через неточності алгоритмів детекції й налаштувань інструменту. У випадку з HSTS відсутність заголовка може бути цілком допустимою конфігурацією, тож хоча ця помилка з’являється в звіті, вона не відображає реальної загрози. Ми свідомо відхиляємо її як несуттєву для нашої системи.

ВИСНОВКИ

У результаті виконання даної дипломної роботи було здійснено детальний аналіз і порівняння основних криптографічних протоколів захисту інформації в мережі Інтернет, зокрема протоколів SSL і TLS. Це дослідження виявилося важливим кроком у розумінні способів підвищення безпеки інформаційних обмінів в мережі Інтернет. Було виявлено, що незважаючи на високий рівень захисту, який надають ці протоколи, існують потенційні вразливості, які можуть бути використані зловмисниками. До них належать слабкі місця у конфігурації вебсервера, неоновлене програмне забезпечення та використання застарілих версій протоколів.

Результати цієї роботи мають велике значення для галузі захисту інформації. Вони допомагають у розумінні, як правильно конфігурувати вебсервери для надання максимального захисту від кібератак. Крім того, рекомендації з покращення безпеки, викладені в цій роботі, можуть бути використані практиками в області комп'ютерної інженерії.

Результати даного дослідження можуть бути використані для покращення існуючих криптографічних протоколів, зокрема SSL і TLS. Детальний аналіз виявив потенційні вразливості цих протоколів та можливі шляхи їх усунення, що може сприяти підвищенню рівня безпеки при передачі інформації через Інтернет. Окрім цього, аналіз та порівняльна оцінка цих протоколів можуть бути використані розробниками та інженерами при модернізації існуючих протоколів.

З урахуванням складності і постійної зміни технологічного середовища, рекомендується продовжити дослідження в цій області. В перспективі особливої уваги вимагає дослідження впливу квантових технологій на криптографічні протоколи. Адже розвиток квантового обчислення може категорично змінити існуючі моделі захисту і, як наслідок, потребуватиме від нас нових стратегій захисту від кібератак.

ПЕРЕЛІК ПОСИЛАНЬ

1. Automated Validation of Internet Security Protocols and Applications (AVISPA) [Електронний ресурс] / AVISPA. – 2003. – Режим доступу: http://www.avispa-project.org/delivs/6.1/d6-1/node3.html – Дата звернення: 08.04.2025.
2. Katz J. Introduction to Modern Cryptography: Third Edition (Chapman & Hall/CRC Cryptography and Network Security Series) / J. Katz, Y. Lindell. – Chapman and Hall/CRC, 2020. – 648 с.
3. Schneier B. Applied Cryptography / Bruce Schneier. – John Wiley & Sons, 1996. – 784 с.
4. Stallings W. Cryptography and Network Security: Principles and Practice (6th Edition) / William Stallings. – Pearson, 2013. – 752 с.
5. Sunshine C. A. Computer Network Architectures and Protocols / Carl A. Sunshine. – Springer Science & Business Media, 2013. – 542 с.
6. Tanenbaum A. S. Computer Networks (6th Edition) / A. S. Tanenbaum, N. Feamster, D. J. Wetherall. – Pearson, 2020. – 960 с.
7. Вербіцький О.В. Вступ до криптології / О. В. Вербіцький. – Львів: Вид-во НТЛ, 2008. – 248 с.
8. Лагун А. Е. Криптографічні системи та протоколи: навч. посіб. / А. Е. Лагун. – Львів: Вид-во Львів. політехніки, 2013. – 96 с.
9. Diffie W., Hellman M. New directions in cryptography // IEEE Transactions on Information Theory. – 1976. – Vol. 22, No. 6. – P. 644–654.
10. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems // Communications of the ACM. – 1978. – Vol. 21, No. 2. – P. 120–126.
11. Головченко С.І. Охорона праці. Основи ергономіки: Підручник / С.І. Головченко, І.А. Косарєв, М.А. Словак. – К.: Вища школа, 2016. – 384 с.
12. НПАОП 0.00-1.28-10 Правила охорони праці при експлуатації електронно-обчислювальних машин. – К., 2010. – 16 с.

13. Singh S. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography / Simon Singh. – Anchor Books, 2000. – 432 с.

14. Ferguson N., Schneier B., Kohno T. Cryptography Engineering: Design Principles and Practical Applications / Niels Ferguson, Bruce Schneier, Tadayoshi Kohno. – John Wiley & Sons, 2010. – 384 с.

15. Rivest R. L., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems / R. L. Rivest, A. Shamir, L. Adleman. – Communications of the ACM, 1978. – 21 (2). – С. 120–126.

16. Stinson D. R. Cryptography: Theory and Practice (3rd Edition) / Douglas R. Stinson. – Chapman and Hall/CRC, 2005. – 616 с.

17. Lucas M. Absolute FreeBSD: The Complete Guide to FreeBSD / M. Lucas. – No Starch Press, 2018. – 720 с.

18. Nessus [Електронний ресурс] / Tenable, Inc. – 2023. – Режим доступу: https://www.tenable.com/products/nessus - Дата звернення: 11.05.2023.

19. Nessus Documentation [Електронний ресурс] / Tenable, Inc. – 2023. – Режим доступу: https://docs.tenable.com/nessus/Content/Welcome.htm - Дата звернення: 10.05.2023.

20. Apache HTTP Server Documentation Version 2.4 [Електронний ресурс] / The Apache Software Foundation. – 2025. – Режим доступу: https://httpd.apache.org/docs/2.4/ - Дата звернення: 17.04.2025.

21. Bowen A., Coar K. Apache Cookbook: Solutions and Examples for Apache Administration / A. Bowen, K. Coar. – O'Reilly Media, 2007. – 252 с.

22. Rescorla E. SSL and TLS: Designing and Building Secure Systems / E. Rescorla. – Addison-Wesley Professional, 2000. – 528 с.

23. Dierks T., Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2 [Електронний ресурс] / T. Dierks, E. Rescorla. – The Internet Engineering Task Force (IETF). – 2008. – Режим доступу: https://tools.ietf.org/html/rfc5246 - Дата звернення: 07.05.2023.

24. FreeBSD Ports Collection: Apache24 [Електронний ресурс]. – The FreeBSD Project. – 2025. – Режим доступу: https://www.freshports.org/www/apache24/ - Дата звернення: 01.05.2025.

ДОДАТОК А
Лістинг модуля шифрування та дешифрування

```
# файл: crypto_module.py

from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes

class HybridCryptoSystem:
    def __init__(self, rsa_key_size=4096):
        # Генерація RSA-ключів
        self._rsa_key = RSA.generate(rsa_key_size)
        self.public_key = self._rsa_key.publickey().export_key()
        self._private_key = self._rsa_key.export_key()

    def encrypt(self, data: bytes) -> dict:
        # 1. Створення сеансового AES-ключа
        session_key = get_random_bytes(32)  # AES-256
        # 2. Шифрування даних AES
        cipher_aes = AES.new(session_key, AES.MODE_GCM)
        ciphertext, tag = cipher_aes.encrypt_and_digest(data)
        # 3. Шифрування AES-ключа RSA-ом
        rsa_cipher = PKCS1_OAEP.new(RSA.import_key(self.public_key))
        enc_session_key = rsa_cipher.encrypt(session_key)
        return {
            'enc_session_key': enc_session_key,
            'nonce': cipher_aes.nonce,
            'tag': tag,
            'ciphertext': ciphertext
        }

    def decrypt(self, enc_data: dict) -> bytes:
        # 1. Розшифрування сеансового ключа RSA-ом
        rsa_cipher = PKCS1_OAEP.new(RSA.import_key(self._private_key))
        session_key = rsa_cipher.decrypt(enc_data['enc_session_key'])
        # 2. Дешифрування даних AES
        cipher_aes = AES.new(session_key, AES.MODE_GCM,
                             nonce=enc_data['nonce'])
        return cipher_aes.decrypt_and_verify(
            enc_data['ciphertext'], enc_data['tag']
        )
```

ДОДАТОК Б
Інструкція користувача з прикладом коду

1. Вимоги до системи
  - ОС: Windows 10+ або Linux (Ubuntu 18.04+)
  - Python 3.9+
  - Бібліотеки: pycryptodome, tkinter

2. Інсталяція
  pip install pycryptodome

3. Запуск програми
  python main.py

Нижче наведено приклад коду для запуску графічного інтерфейсу на Tkinter:

```
import tkinter as tk
from crypto_module import HybridCryptoSystem

def encrypt_file():
    path = file_entry.get()
    with open(path, 'rb') as f:
        data = f.read()
    crypto = HybridCryptoSystem()
    enc = crypto.encrypt(data)
    with open(path + '.enc', 'wb') as out:
        out.write(enc['enc_session_key'] + enc['nonce'] + enc['tag'] +
enc['ciphertext'])
    status_label.config(text='Файл зашифровано')

def decrypt_file():
    path = file_entry.get()
    with open(path, 'rb') as f:
        raw = f.read()
    crypto = HybridCryptoSystem()
    # Розпакувати enc_session_key, nonce, tag, ciphertext згідно
формату
    # ...
    status_label.config(text='Файл розшифровано')

root = tk.Tk()
root.title('Crypto GUI')
file_entry = tk.Entry(root, width=50)
file_entry.pack(padx=10, pady=10)
tk.Button(root, text='Шифрувати',
command=encrypt_file).pack(side='left', padx=5)
tk.Button(root, text='Розшифрувати',
command=decrypt_file).pack(side='right', padx=5)
status_label = tk.Label(root, text='')
```

```
status_label.pack(pady=10)
root.mainloop()
```