Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: " Методи шифрування даних в операційних системах. Unix та MacOS"

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Навроцький Д.П.
підпис (прізвище та ініціали)

Керівник Загородна Н.В.
підпис (прізвище та ініціали)

Нормоконтроль Лобур Т.Б.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Навроцькому Дмитру Петровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи шифрування даних в операційних системах Unix та MacOS

Керівник роботи Загородна Наталія Володимирівна, к.т.н., професор кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «03» 04 2023 року № 4/7-349

2. Термін подання студентом завершеної роботи 14.06.2023

3. Вихідні дані до роботи Вимоги до операційних систем Unix та MacOS

4. Зміст роботи (перелік питань, які потрібно розробити)
Огляд загальних принципів шифрування, включаючи базові поняття та класифікацію методів шифрування.
Розглянути загальні принципи шифрування даних операційних системах Unix та MacOS
Розглянути методи шифрування даних в операційній системі FreeBSD
Проаналізувати модуль шифрування GELI (GEOM-based Encryption Layer) та шифровану файлову систему EncFS
Розглянути методи шифрування даних в операційній системі MacOS
Проаналізувати вбудований механізм шифрування диску FileVault
Проаналізувати можливості шифрування даних за допомогою Encrypted APFS
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, мета, задачі. Огляд криптографічних методів та функцій. Симетричне шифрування. Асиметричне шифрування. Хеш-функції. Гібридне шифрування. Огляд можливостей шифрування в операційній системі FreeBSD. Процедура шифрування даних за допомогою GILE. Процедура шифрування даних за допомогою EncFS. Можливості шифрування в операційній системі MacOS. FileVault шифрування. Шифрування жорсткого диску за допомогою Disk Utility та Encrypted APFS. Переваги та недоліки GELI в FreeBSD. Переваги та недоліки EncFS в FreeBSD. Переваги та недоліки Encrypted APFS в MacOS. Сумісність методів шифрування. Відкриття зашифрованого файлу в FreeBSD на MacOS. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Пилипець М.І., проф. кафедри МТ		

7. Дата видачі завдання 22.01.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	22.01 – 24.01	Виконано
2.	Підбір джерел аналізу роботи шифрування в Unix та MacOS	25.01 – 29.01	Виконано
3.	Опрацювання джерел в галузі дослідження	05.02 – 17.02	Виконано
4.	Проаналізовати модуль шифрування GELI (GEOM-based Encryption Layer) та шифровану файлову систему EncFS	07.03 – 22.03	Виконано
5.	Проаналізовати вбудований механізм шифрування диску FileVault та Encrypted APFS в MacOS	24.03-12.04	Виконано
6.	Оформлення розділу «Огляд криптографічних методів та функцій»	18.03 – 25.03	Виконано
7.	Оформлення розділу «Шифрування даних в операційних системах»	26.04 – 05.05	Виконано
8.	Оформлення розділу «Сумісність методів шифрування, переваги та недоліки»	06.05 – 15.05	
9.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	16.05 – 26.05	Виконано
10.	Оформлення кваліфікаційної роботи	27.05 – 06.06	Виконано
11.	Нормоконтроль	08.06 – 14.06	Виконано
12.	Перевірка на плагіат	11.06 – 12.06	Виконано
13.	Попередній захист кваліфікаційної роботи	1406 – 15.06	Виконано
14.	Захист кваліфікаційної роботи	23.06.2023	

Студент _____ Навроцький Д.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи шифрування даних в операційних системах Unix та MacOS// Кваліфікаційна робота ОР «Бакалавр» // Навроцький Дмитро Петрович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-41 // Тернопіль, 2023 // С. 62 , рис. –47, табл. – __ , кресл. 23 , додат. – ___.

Ключові слова: ШИФРУВАННЯ, ОПЕРАЦІЙНІ СИСТЕМИ, UNIX, FREEBSD, MACOS, APFS, GELI, ENCFS.

Кваліфікаційна робота присвячена аналізу методів шифрування даних в Unix та MacOS. Дослідження в цій роботі дозволяє зрозуміти принципи роботи з шифруванням даних в цих операційних системах, оцінити їх ефективність, переваги та недоліки.

Огляд загальних принципів шифрування, включаючи поняття та класифікацію методів шифрування розглянуто в першому розділі кваліфікаційної роботи.

В другому розділі розглянуто загальні принципи шифрування даних в операційній системі FreeBSD, як представнику операційних систем типу Unix. Показано методику шифрування файлів, шифрування дискового простору. Проаналізовано методи шифрування за допомогою GELI та EncFS. Розглянуто принципи роботи цих методів, їх застосування. Розглянуто методи шифрування даних в операційній системі MacOS. Проаналізовано методи Шифрування жорсткого диску в цілому за допомогою Disk Utility та Encrypted APFS та створення зашифрованого дискового образу за допомогою Disk Utility з файловою системою Encrypted APFS.

В третьому розділі проведено порівняльний аналіз різних методів шифрування даних в Unix та MacOS, включаючи їх переваги, недоліки, рівень захисту, зручність використання та інші фактори, що впливають на вибір методу шифрування. Даний розділ також включає аналіз сумісності між різними методами шифрування.

ANNOTATION

Data encryption methods in Unix and MacOS operating systems // Thesis of educational level "Bachelor" // Dmytro Navrotskyi // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, СБс-41 group // Ternopil, 2023 // P. _62__ , fig. -_47___, table. - __ , chair. - _23__ , added. -___.

Keywords: ENCRYPTION, OPERATING SYSTEMS, UNIX, FREEBSD, MACOS, APFS, GELI, ENCFS.

The qualification thesis is devoted to the analysis of data encryption methods in Unix and MacOS. The research in this paper allows understanding the principles of data encryption in these operating systems, evaluating their effectiveness, advantages, and disadvantages. The work can be used for educational purposes to illustrate encryption methods in Unix and MacOS.

An overview of general encryption principles, including the concepts and classification of encryption methods, is considered in the first chapter of the qualification paper.

The second chapter discusses the general principles of data encryption in the FreeBSD operating system as a representative of Unix-like operating systems. It presents the methodology of file encryption, disk space encryption. The encryption methods using GELI and EncFS are analyzed. The data encryption methods in the MacOS operating system are examined. The encryption methods of the hard disk as a whole using Disk Utility and Encrypted APFS, and the creation of an encrypted disk image using Disk Utility with the Encrypted APFS file system are discussed. The principles of operation and application of these methods are examined.

The third chapter provides a comparative analysis of various data encryption methods in Unix and MacOS, including their advantages, disadvantages, level of protection, usability, and other factors influencing the choice of encryption method. This chapter also includes an analysis of compatibility between different encryption methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GELI — GEneric OS Module ELI
APFS — Apple File System
DES — Data Encryption Standard
AES — Advanced Encryption Standard
CBC — Cipher Block Chaining
ECB — Electronic Codebook
XTS — XEX-based Tweaked Codebook Mode with Cipher-Text Stealing
RSA — Rivest Shamir Adleman
ECC — Elliptic Curve Cryptography
MD5 — Message Digest Algorithm 5
SHA-1 — Secure Hash Algorithm 1
SHA-256 — Secure Hash Algorithm 256
SHA-512 — Secure Hash Algorithm 512
BSD — Berkeley Software Distribution
GELI — GEOM-based Encryption Layer
PBKDF2 — Password-Based Key Derivation Function 2
HMAC — Keyed-Hash Message Authentication Code
SSL — Secure Sockets Layer
SFTP — Secure File Transfer Protocol
SSH — Secure Shell

ВСТУП

Безпека даних є критичним фактором у світі, де кількість цифрових загроз постійно зростає. Шифрування – це процес перетворення звичайного тексту або даних у незрозумілу форму (шифр) за допомогою спеціального алгоритму. Це важливий механізм для захисту конфіденційності, цілісності та автентичності інформації. Шифрування дозволяє забезпечити безпеку даних шляхом утруднення доступу до них неповноважним особам. Зашифровані дані можуть бути розшифровані лише з використанням правильного ключа або пароля. Шифрування застосовується у різних сферах, включаючи комунікації, зберігання даних та обробку інформації.

В роботі будуть розглянуті основні принципи та характеристики кожного методу шифрування, їх переваги та недоліки. Також буде проведений аналіз ефективності та надійності кожного з них. Основна мета кваліфікаційної роботи полягає у дослідженні ефективності та надійності методів шифрування даних в операційних системах Unix та MacOS для захисту важливої інформації в сучасному цифровому середовищі.

У рамках кваліфікаційної роботи будуть розглянуті дві популярні операційні системи – FreeBSD, як представник операційних систем типу Unix і MacOS Monterey, оскільки вони широко використовуються як у корпоративному, так і в особистому середовищі. Перш за все, буде проаналізовано метод шифрування даних GELI в операційній системі FreeBSD. GELI є модулем ядра, який надає можливість створення шифрованих контейнерів даних і шифрування цілих розділів на рівні блоків. Дослідження включатиме вивчення принципів роботи GELI, його конфігурацію, можливості та обмеження.

Другий метод, що буде розглянутий, - EncFS. EncFS є файловою системою, яка працює на рівні користувача і забезпечу шифрування окремих файлів або директорій. EncFS пропонує простий у використанні інтерфейс, але водночас забезпечує надійний рівень шифрування даних. У кваліфікаційній роботі буде проведений аналіз алгоритму шифрування та можливостей EncFS.

В операційній системі MacOS буде розглянуто два методи шифрування даних. Перший - FileVault, вбудований інструмент шифрування диска в MacOS. Він надає можливість шифрування всього системного диска або окремих розділів з використанням стандарту XTS-AES.

Також буде розглянуто Encrypted APFS. Це нова файлова система, впроваджена в MacOS High Sierra і новіших версіях, яка надає вбудовану підтримку шифрування даних. Encrypted APFS використовує сучасні криптографічні алгоритми та механізми шифрування для забезпечення безпеки даних на рівні файлової системи. У кваліфікаційній роботі буде проведений аналіз принципів роботи Encrypted APFS, показано методику шифрування.

Буде проведено аналіз сумісності між різними методами шифрування даних в Unix та MacOS. Також буде досліджено переваги, недоліки, рівень захисту, зручність використання різних методів шифрування даних.

1 ОГЛЯД КРИПТОГРАФІЧНИХ МЕТОДІВ ТА ФУНКЦІЙ

Криптографічні методи та функції використовуються для захисту конфіденційності, цілісності та автентичності даних. Існує кілька різних криптографічних методів, які можуть бути класифіковані за різними ознаками, такими як використання ключів, алгоритми, кількість ключів та інші [1].

1.1 Симетричне шифрування

Симетричне шифрування - це вид криптографічного шифрування, де той самий ключ використовується як для шифрування, так і для розшифрування повідомлень. Це означає, що як відправник, так і отримувач повинні знати (мати) один і той же ключ для захисту та розшифрування повідомлення [2].

Одна з основних переваг симетричного шифрування - його швидкодія. Симетричні шифри виконують шифрування та розшифрування дуже швидко, оскільки вони використовують прості математичні операції, такі як побітові операції, що дозволяє їм бути ефективними в обчислювальному плані.

Проте, головний недолік симетричного шифрування - це потреба в передачі та зберіганні спільного ключа між відправником та отримувачем. Якщо цей ключ потрапляє в руки зловмисника, то це може призвести до розкриття всіх зашифрованих повідомлень, які використовують цей ключ.

Одним з основних аспектів симетричного шифрування є вибір правильного ключа. Ключ повинен бути достатньо довгим та складним, щоб ускладнити спроби зламати шифр. Однак, однаково, як і з ростом довжини ключа, збільшується його складність, так і збільшується вимога до передачі та зберігання такого ключа.

Одна з важливих викликів симетричного шифрування - це безпека ключа. Якщо ключ втрачається або стає відомим стороннім особам, то всі зашифровані дані, захищені цим ключем, можуть бути скомпрометовані.

Існує також питання обмеженості використання ключа. У симетричному шифруванні кожна пара користувачів повинна мати свій власний спільний ключ,

що може бути неефективним при великій кількості користувачів або в випадку потреби в захисті багатьох різних комунікацій.

Симетричне шифрування не надає механізмів для ідентифікації відправника даних. Це може створювати вразливості.

Незважаючи на ці виклики, симетричне шифрування залишається важливим і ефективним засобом захисту даних. Воно широко використовується в багатьох випадках, де вимога до швидкодії та ефективності переважає недоліки передачі та зберігання спільного ключа. Загалом, вибір між симетричним та асиметричним шифруванням залежить від конкретних потреб захисту даних та вимог до безпеки, зручності використання та ефективності.

Приклади симетричних шифрів включають DES, AES та Blowfish.

DES - цей шифр використовується для захисту конфіденційної інформації. Він став першим широко використовуваним стандартом шифрування, який прийняла урядова організація США у 1977 році. DES використовує 56-бітний ключ для шифрування повідомлення та 64-бітний блок. Triple DES - це удосконалена версія DES, яка використовує три ключі замість одного.

AES - цей шифр є більш сучасним стандартом шифрування, який прийнято у 2001 році. AES використовує 128-, 192- або 256-бітні ключі для шифрування повідомлень та 128-бітний блок. Він зараз є стандартом у багатьох криптографічних протоколах, таких як SSL, TLS та SSH. Також він є стандартом для шифрування даних в операційних системах Windows, Linux, Unix та MacOS.

Blowfish - цей шифр був розроблений у 1993 році. Він використовується для шифрування даних, паролів та інших конфіденційних даних. Blowfish використовує 64-бітний блок і ключ, який може бути до 448 біт. Хоча Blowfish зараз вважається менш безпечним, ніж AES, але він залишається досить популярним у багатьох додатках.

Потокове та блочне шифрування є двома різними підходами до криптографічного захисту інформації в симетричному шифрування.

Потокове шифрування - це метод шифрування, при якому дані розбиваються на окремі біти або байти, і кожен біт або байт шифрується окремо за допомогою шифрувального алгоритму, який генерує послідовність

псевдовипадкових бітів, відомих як "потік". Ці псевдовипадкові біти використовуються для комбінування з вхідними даними за допомогою операції логічного XOR для отримання зашифрованих даних. Потокове шифрування забезпечує швидку обробку даних, оскільки вони можуть бути шифровані одночасно біт за бітом. Однак, важливо враховувати безпеку генерації потоку псевдовипадкових бітів, оскільки це визначає безпеку шифрування в цілому.

Прикладом потокових шифрів є RC4, Salsa20 та ChaCha20, які широко використовуються в різних додатках.

Блочне шифрування - це метод шифрування, при якому дані розбиваються на рівні блоки фіксованого розміру, наприклад 64 або 128 бітів, і кожен блок шифрується окремо за допомогою шифрувального алгоритму. В блочному шифруванні кожен блок вхідних даних замінюється на блок зашифрованих даних з використанням ключа шифрування. Блочне шифрування забезпечує високий рівень безпеки, оскільки зміна одного біта в вхідних даних призводить до зміни всього зашифрованого блоку. Однак, блочне шифрування може бути менш ефективним для обробки великих обсягів даних, оскільки блоки потребують додаткової обробки.

Кожен блок може бути залежним від попередніх блоків в режимі CBC або може бути незалежним в режимі ECB. Також може бути використано режим XTS та інші.

Прикладом потокових шифрів є AES, DES, Triple DES та Blowfish, які також є популярними в різних застосунках.

1.2 Асиметричне шифрування

Асиметричне шифрування, також відоме як криптографія з відкритим ключем, це метод шифрування, в якому використовується пара ключів: приватний ключ і публічний ключ [3]. Цей підхід використовується для захисту конфіденційності та цілісності даних.

Приватний ключ є секретним ключем, відомим тільки власнику, і використовується для розшифрування даних або підпису повідомлень.

Публічний ключ є відкритим ключем, доступним для всіх, і використовується для шифрування даних або перевірки підпису повідомлень.

Одна з основних переваг асиметричного шифрування полягає в тому, що не потрібно передавати приватний ключ між відправником та отримувачем, що робить цей метод більш безпечним у порівнянні з симетричним шифруванням, де використовується один спільний ключ для шифрування та розшифрування даних. Асиметричне шифрування також використовується для цифрових підписів, які дозволяють перевірити автентичність повідомлень та ідентифікацію відправника.

Однак, асиметричне шифрування також має свої обмеження. Воно може бути повільнішим та менш ефективним з точки зору обчислювальних ресурсів, ніж симетричне шифрування, через використання більших ключів та складніших алгоритмів. Тому, зазвичай використовують комбінацію асиметричного та симетричного шифрування, де асиметричне шифрування використовується для обміну симетричним ключем, який потім використовується для шифрування реальних даних.

Асиметричне шифрування має широке застосування в сучасних системах безпеки, таких як захищені комунікації в Інтернеті (наприклад, захищений протокол передачі даних SSL/TLS), електронний підпис, аутентифікація користувачів, забезпечення конфіденційності даних та багато інших застосувань, де важлива безпека та захист інформації.

Приклади асиметричних шифрів включають RSA та ECC.

RSA - це один з найпоширеніших асиметричних криптографічних алгоритмів, який використовується для шифрування та підпису даних. Основна ідея RSA полягає в використанні пари ключів - приватного та публічного. Приватний ключ використовується для розшифрування даних та підпису, тоді як публічний ключ використовується для шифрування даних та перевірки цифрових підписів.

Довжина ключа RSA визначається кількістю бітів в числовому значенні, яке використовується для генерації ключів. Більша довжина ключа забезпечує більш високий рівень безпеки, але також може призводити до більшого

обчислювального навантаження під час виконання операцій шифрування та розшифрування.

Загальноприйняті рекомендації щодо довжини ключа RSA змінюються з часом в залежності від зростання обчислювальної потужності атакуючих систем. На сьогоднішній день рекомендовані мінімальні довжини ключа RSA для різних застосувань такі:

– 2048 біт - рекомендований мінімальний рівень безпеки для більшості додатків, зокрема для захищених комунікацій в Інтернеті;

– 3072 біт і більше - рекомендований рівень безпеки для додатків, де ключі можуть використовуватися тривалий час, таких як захищені зв'язки державного рівня;

– 4096 біт і більше - рекомендований рівень безпеки для високочутливих додатків, де вимагається найвищий рівень захисту, таких як фінансові та військові системи.

Проте, варто враховувати, що більша довжина ключа також призводить до більшого обчислювального навантаження при виконанні операцій шифрування та розшифрування, що може вплинути на продуктивність системи. Тому вибір довжини ключа RSA повинен бути зроблений як компроміс між безпекою та продуктивністю з урахуванням конкретних потреб та вимог застосування.

ECC - це криптографічний алгоритм, який використовує еліптичні криві для забезпечення захисту інформації. В порівнянні з традиційними криптографічними алгоритмами, такими як RSA, ECC забезпечує еквівалентний рівень безпеки з меншою довжиною ключа, що робить його особливо підходящим для обмежених ресурсів, таких як вбудовані системи та мобільні пристрої.

ECC забезпечує еквівалентний рівень безпеки до RSA з меншою довжиною ключа. Наприклад, ключ ECC довжиною 256 біт забезпечує безпеку, що еквівалентна ключу RSA довжиною 2048 біт.

ECC вимагає меншої обчислювальної потужності, меншої кількості пам'яті і меншого обсягу даних для передачі порівняно з іншими криптографічними

алгоритмами, такими як RSA, що робить його більш ефективним для реалізації на пристроях з обмеженими ресурсами.

Операції шифрування, розшифрування, підпису та перевірки підпису на еліптичних кривих можуть бути виконані швидше, ніж на RSA з аналогічним рівнем безпеки. ECC може бути використаний для різних криптографічних операцій, таких як шифрування, розшифрування, підписування, перевірки підпису, обмін ключами та інші криптографічні операції. Він може бути використаний в різних додатках, включаючи захист комунікацій в мережах, електронну комерцію, мобільні додатки, вбудовані системи, Інтернет речей (IoT) та інші.

Довжина ключа ECC (еліптичної криптографії) вимірюється в бітах і відрізняється від RSA. Для ECC використовуються ключі, представлені у вигляді точок на еліптичній кривій, і довжина ключа визначається кількістю бітів, необхідних для представлення координат цих точок.

Типові розміри ключів ECC включають 128 біт, 192 біта, 256 біт, 384 біти, 521 біт.

Довжина ключа ECC обирається залежно від рівня безпеки, який ви хочете забезпечити. Загалом, більша довжина ключа забезпечує вищий рівень безпеки, але може вимагати більших обчислювальних ресурсів для операцій шифрування/розшифрування, підпису/перевірки підпису тощо.

1.3 Хеш-функції

Хеш-функції (також відомі як криптографічні хеш-функції) є однією з основних криптографічних технік, використовуваних для забезпечення захисту даних в криптографії та інформаційній безпеці [4]. Хеш-функції приймають вхідні дані довільної довжини і генерують фіксований вихідний рядок фіксованої довжини, який часто представляється у вигляді хеш-коду або дайджесту.

Основна властивість хеш-функцій - відсутність зворотної дії, тобто неможливість відновлення вихідних даних з хеш-коду без знання вхідних даних. Це робить хеш-функції придатними для використання при забезпеченні

цілісності даних та перевірки на подібність (наприклад, перевірки цілісності файлів), а також для забезпечення аутентифікації та контролю доступу (наприклад, для зберігання паролів у захешованому вигляді).

Одна з важливих властивостей криптографічних хеш-функцій - стійкість до колізій, тобто важкість знайти два різні вхідні повідомлення, які мають однаковий хеш-код. Це робить можливим використання хеш-функцій для створення цифрових підписів, електронного голосування, створення блокчейнів та багатьох інших додатків, де важлива недоступність колізій.

Проте важливо зазначити, що хеш-функції не є універсальним засобом захисту даних, і вони можуть мати певні обмеження та уразливості. Для криптографічних додатків рекомендується використовувати спеціальні криптографічні хеш-функції, які відповідають сучасним стандартам безпеки, і використовувати їх у комбінації з іншими криптографічними методами для забезпечення надійності та безпеки даних.

Деякі приклади відомих криптографічних хеш-функцій включають MD5 SHA-1, SHA-256 , SHA-512, SHA-3-256 та SHA-3-512.

1.4 Гібридне шифрування

Гібридне шифрування - це метод шифрування, який комбінує переваги симетричного та асиметричного шифрування для забезпечення безпеки і конфіденційності даних [5]. У гібридному шифруванні використовуються два різні типи ключів: симетричний ключ та публічний ключ.

Симетричне шифрування використовує один ключ для шифрування та розшифрування даних. Однак використання одного ключа має потенційні ризики, такі як безпека ключа та необхідність обміну ключами між відправником та одержувачем.

Асиметричне шифрування використовує два ключі - публічний ключ та приватний ключ. Публічний ключ використовується для шифрування даних, тоді як приватний ключ використовується для розшифрування даних. Це дозволяє

безпечно обмінюватися публічним ключем, оскільки він не розкриває приватний ключ.

Гібридне шифрування використовує обидва ці підходи: спочатку дані шифруються симетричним ключем, а потім цей симетричний ключ шифрується публічним ключем одержувача. Після отримання зашифрованих даних одержувач використовує свій приватний ключ для розшифрування симетричного ключа, а потім використовує цей симетричний ключ для розшифрування самої інформації.

Гібридне шифрування дозволяє поєднати ефективність симетричного шифрування з безпекою асиметричного шифрування.

Гібридне шифрування широко використовується в різних застосуваннях, таких як захищений обмін електронною поштою. Приклад такої системи - Pretty Good Privacy), яка використовує гібридне шифрування для захисту електронної пошти. Також гібридне шифрування використовується в захищених з'єднаннях SSL/TLS в Інтернеті, електронних платежах та інші випадки, де безпека та конфіденційність даних є важливими.

2 ШИФРУВАННЯ ДАНИХ В ОПЕРАЦІЙНИХ СИСТЕМАХ

2.1 Операційна система FreeBSD

FreeBSD - це операційна система на базі UNIX, яка розробляється та підтримується FreeBSD Foundation та групою волонтерів.

Назва операційної походить від BSD, яка є операційноє систем на базі UNIX. Операційна система FreeBSD відома своєю стабільністю, продуктивністю та безпекою. FreeBSD має велику кількість різноманітного програмного забезпечення та додатків, які можуть бути встановлені з допомогою системи управління пакетами.

Операційна система також відома своєю високою мірою сумісності з програмним забезпеченням, розробленим для інших UNIX-подібних систем, тому її часто використовують у великих корпоративних середовищах, веб-серверах, маршрутизаторах та інших мережевих пристроях.

FreeBSD є вільною та відкритою операційною системою, що означає, що вона постійно розвивається та підтримується групою волонтерів, а також вона доступна для використання та модифікації безкоштовно.

2.1.1 Огляд можливостей шифрування в операційній системі FreeBSD

У операційній системі FreeBSD є кілька можливостей для шифрування даних.

GELI - це шифрування для блочних пристроїв в операційній системі FreeBSD. Даний метод шифрування дозволяє створювати зашифровані томи та розділи на фізичних дисках, що забезпечує захист від несанкціонованого доступу. GELI забезпечує шифрування на рівні блоків даних, та дозволяє захистити дані навіть у випадку, якщо хакер або зловмисник має фізичний доступ до носія даних. Ключ шифрування зберігається в оперативній пам'яті, що дозволяє швидко шифрувати та розшифровувати дані без затримок. Підтримуються різні алгоритми шифрування, такі як AES, Blowfish, Camellia та

3DES. Крім того, GELI підтримує різні режими роботи цих алгоритмів, такі як CBC, XTS.

Підтримується можливість створення зашифрованих контейнерів, які можуть бути підключені до системи файлів як окремі розділи.

GELI є досить простою та зручною у використанні. Для створення та підключення зашифрованого об'єкту не потрібно використовувати спеціальні інструменти або складні команди. Всі необхідні операції можна виконати за допомогою стандартних засобів операційної системи FreeBSD.

Однак, при використанні GELI потрібно враховувати певні обмеження. Наприклад, шифрування на рівні блоків може призвести до деякого падіння продуктивності системи. Крім того, збереження паролю доступу до зашифрованого об'єкту в оперативній пам'яті може бути небезпечним у випадку, якщо хакер або зловмисник отримає доступ до цієї пам'яті.

У будь-якому випадку, GELI є потужним та ефективним інструментом для захисту даних в операційній системі FreeBSD. Вона може бути використана як для захисту конфіденційних даних на домашньому комп'ютері, так і для захисту даних у великих корпоративних мережах [6].

Також в FreeBSD є можливість здійснити шифрування окремих файлів або тек за допомогою EncFS [7]. Дана програма забезпечує можливість створення віртуального зашифрованого тома на основі існуючої файлової системи. EncFS був розроблений для UNIX-подібних операційних систем і доступний для використання в FreeBSD.

EncFS працює на принципі шифрування файлів під час їх запису та дешифрування під час їх читання. Він забезпечує захист від прямого доступу до файлів на рівні файлової системи, а також від відновлення вмісту файлів після видалення їх з віртуального тому.

Оскільки EncFS використовує шифрування на рівні файлів, його ефективність може бути нижчою, ніж у програм, які використовують шифрування на рівні блоків. EncFS підтримує різні алгоритми шифрування, такі як AES, Blowfish, Camellia. За замовчуванням EncFS підтримує CBC режим роботи цих алгоритмів.

2.1.2 Процедура шифрування даних за допомогою GILE

Щоб створити зашифрований розділ з GELI, спочатку необхідно створити фізичний розділ на диску, який буде використовуватися для зберігання даних. Також можна використати весь диск без створення фізичного розділу. Після цього за допомогою команди geli init створюється зашифрований об'єкт, який пов'язується з фізичним розділом або диском. В процесі створення зашифрованого об'єкту необхідно вказати алгоритм шифрування та пароль для доступу до даних [8].

Після створення зашифрованого об'єкту, його можна відкрити за допомогою команди geli attach, яка підключає об'єкт до системи файлів та забезпечує доступ до даних. Ключ шифрування зберігається в оперативній пам'яті та автоматично видаляється при відключенні об'єкта.

Для створення зашифрованого диску з шифруванням AES 256 за допомогою GELI в операційній системі FreeBSD необхідно виконати такі кроки:

1) Визначити назву диску для шифрування. У даному випадку це буде тестовий диск /dev/da5 (Рис. 2.1).

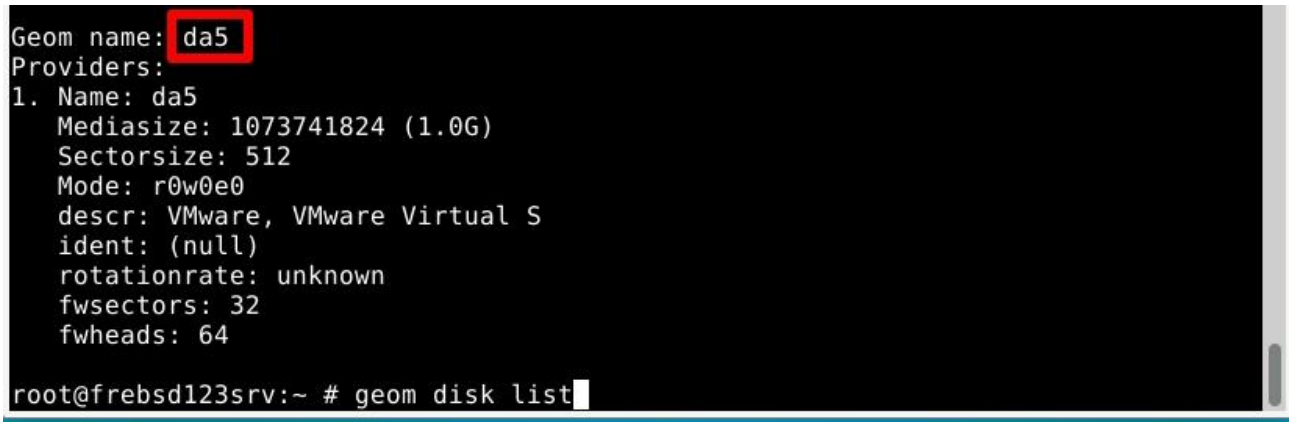

Рисунок 2.1 – Вивід команди geom disk list

2) В терміналі ввести наступну команду:

```
# geli init -b -s 4096 -e aes -l 256 /dev/da5
```

У цій команді:

- `geli init` - ініціалізація GELI диску;
- `-s 4096` - розмір блоку (в байтах), який буде зашифровуватися;
- `-e aes` - використовувати шифрування AES;
- `-l 256` - довжина ключа шифрування (256 біт).

Після введення цієї команди, система запитає пароль доступу до зашифрованого диску (Рис. 2.2).

Рисунок 2.2 – Ініціалізація GELI диску

Після введення пароля, GELI створить ключ шифрування та збереже його в оперативній пам'яті.

3) Для того, щоб здійснити приєднання шифрованого тому, який був створений з використанням GELI, до системи, необхідно ввести наступну команду:

```
# geli attach /dev/da5
```

Команда передбачає, що шифрований том знаходиться на пристрої /dev/da5. Після виконання команди, GELI відкриє шифрований том, з'єднає його з заданим ім'ям пристрою і зробить доступним для використання в системі. Після цього можна працювати з файловою системою, яка міститься на шифрованому томі.

Після введення цієї команди, система запитає пароль доступу до зашифрованого диску. Введіть пароль та підтвердіть його (Рис. 2.3).

Рисунок 2.3 – Приєднання GELI диску

Після введення пароля, GELI ініціалізує зашифрований диск та зробить його доступним для запису та читання.

4) Процес створення файлової системи та монтування шифрованого диску в теку /private показано на рисунку 2.4.

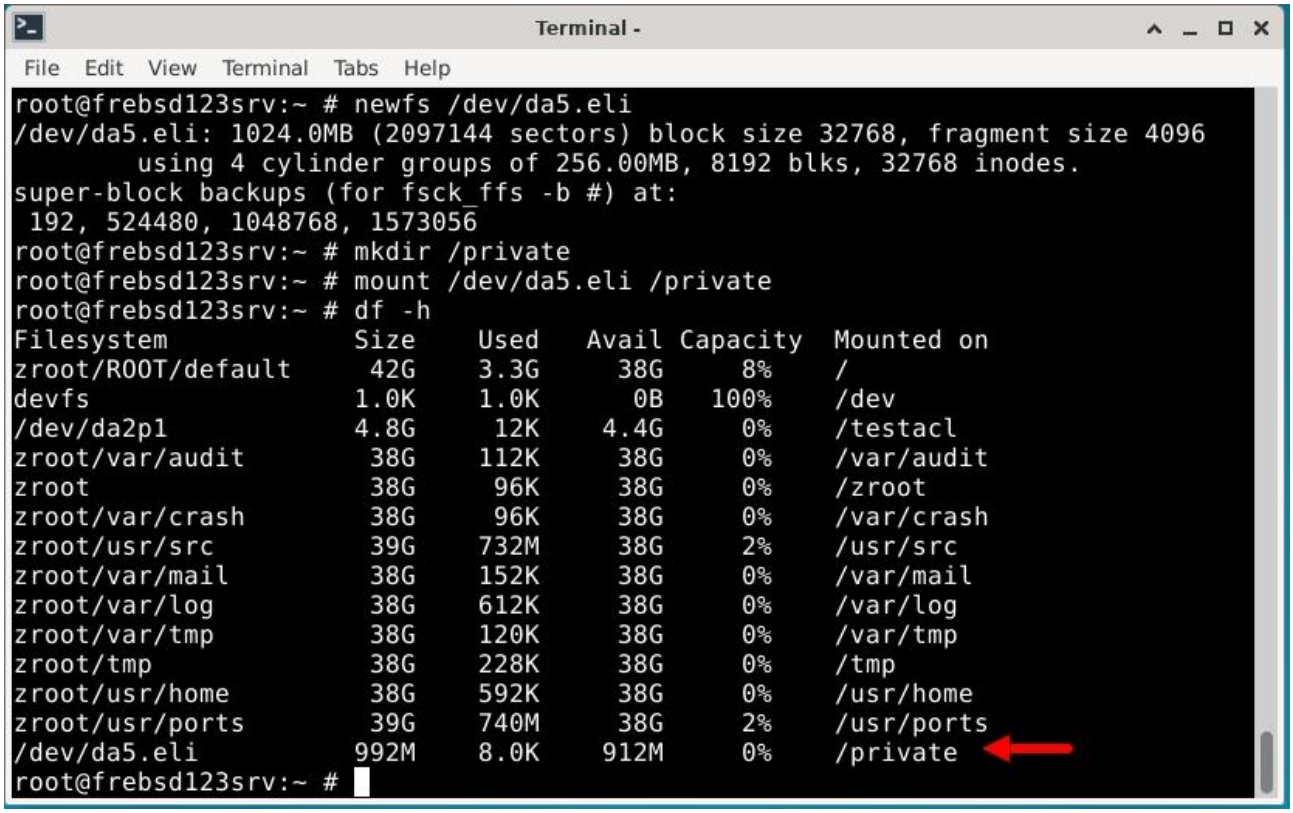

Рисунок 2.4 – Створення файлової системи та монтування шифрованого диску

Тепер диск зашифровано за допомогою GELI та змонтовано в системи.

Щоб подивитись параметри шифрування диску, який зашифрований за допомогою GELI на операційній системі FreeBSD потрібно використати команду `geli list` або `geli dump /dev/da5`.

За допомогою команди `geli list` можна вивести перелік всіх активних GELI об'єктів в системі. Вона виводить інформацію про кожен об'єкт, таку як ім'я об'єкту, стан шифрування, розмір та шифрувальний алгоритм, використовуваний для захисту даних (Рис. 2.5).

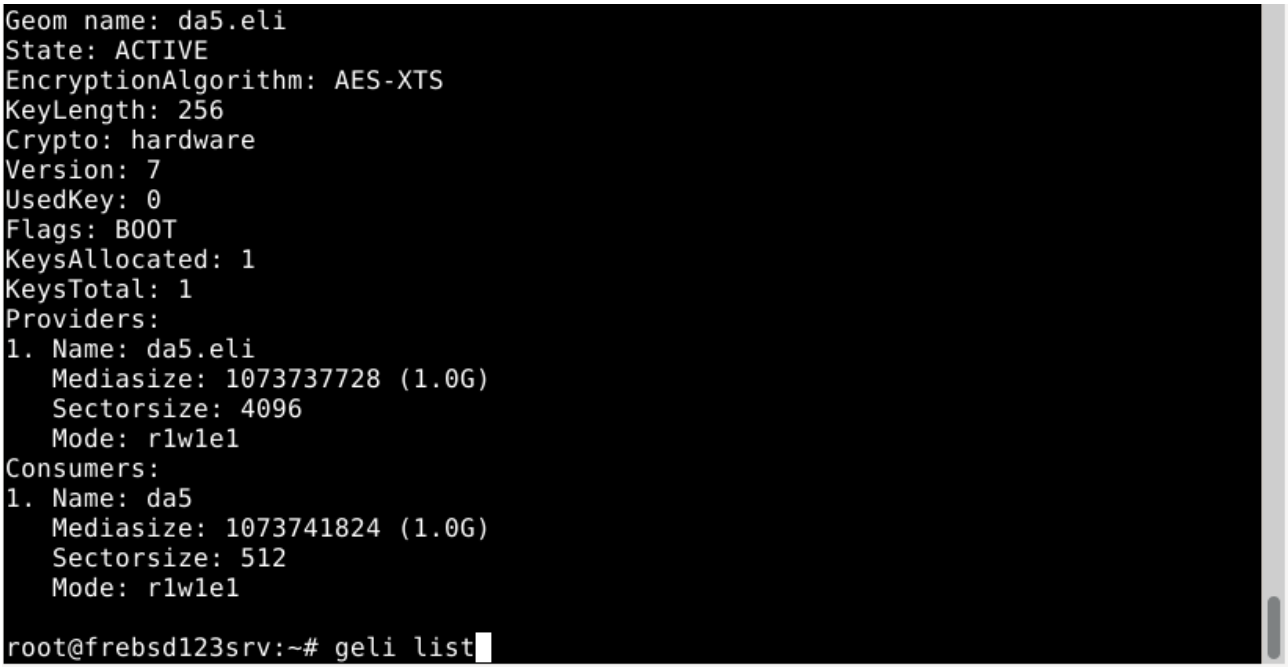

Рисунок 2.5 – Вивід команди geli list

Команда `geli dump /dev/da5` (Рис. 2.6) виводить на екран інформацію про зашифрований об'єкт, який було створено за допомогою GELI. Ця команда використовується для перевірки налаштувань і стану зашифрованого об'єкту, таких як тип шифрування, розмір ключа шифрування, обсяг та розташування секторів, що були зашифровані.

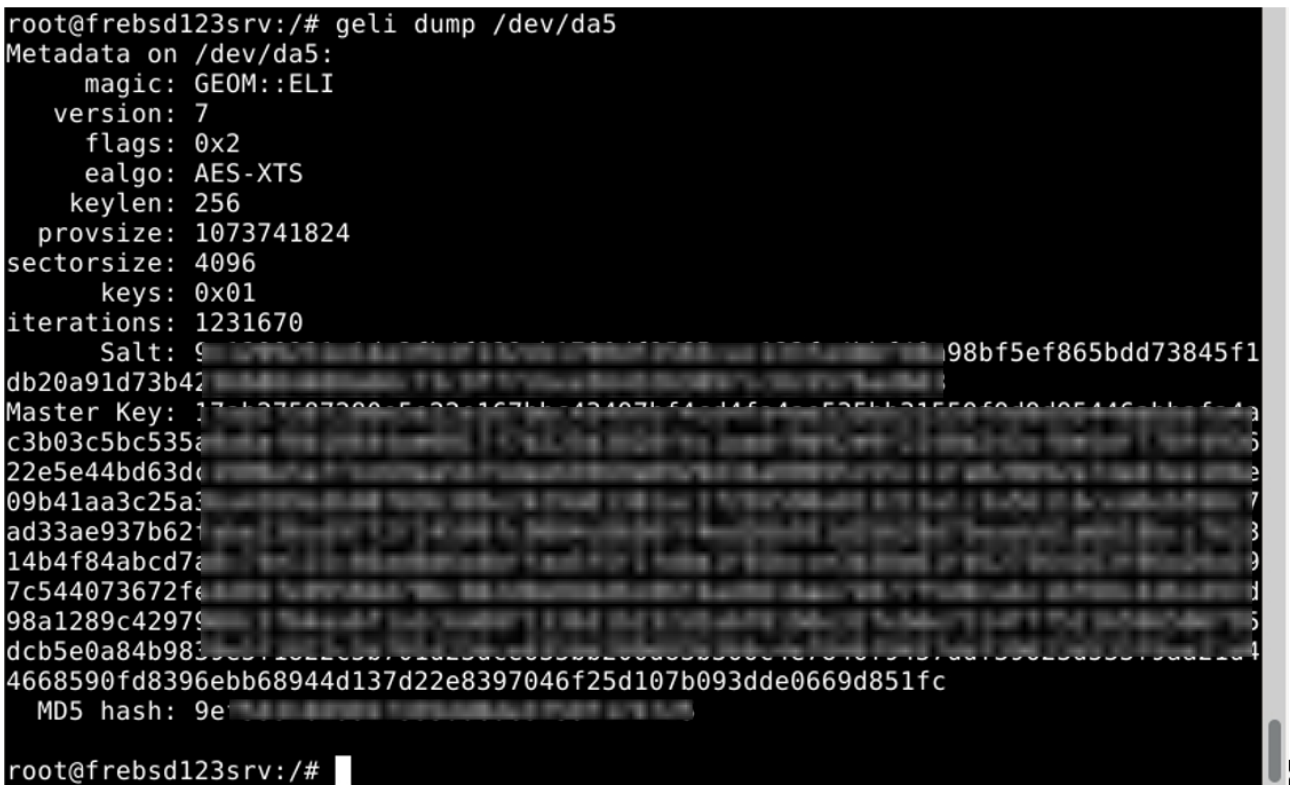

Рисунок 2.5 – Вивід команди geli dump

У даному прикладі, параметри шифрування диску можна розшифрувати наступним чином:

– `ealgo` - це алгоритм шифрування, який використовується для захисту даних на диску. У цьому випадку використовується режим шифрування XTS, який є розширенням алгоритму AES. Він забезпечує кращий захист від атак;

– `keylen` - це довжина ключа шифрування. В цьому випадку ключ складається з 256 біт (32 байти), що забезпечує дуже високий рівень захисту;

– `provsize` - це розмір захищеного простору на диску. В цьому випадку це 1 ГБайт (1073741824 байти);

– `sectorsize` - це розмір сектора на диску. В цьому випадку кожен сектор має розмір 4096 байт;

– `keys` це номер ключа шифрування. В цьому випадку використовується лише один ключ (0x01);

– `iterations` - це кількість ітерацій, які використовувалися для генерації ключа шифрування (1231670). Це значення може бути різним в залежності від налаштувань;

– `Salt` - це випадково згенерований рядок, який використовується як "сіль" для генерації ключа шифрування. Вона додається до парольної фрази перед генерацією Master Key, щоб ускладнити атаки з використанням таблиць зламу паролів. Це забезпечує додатковий рівень захисту від атак;

– `Master Key` - це основний ключ, який використовується для шифрування та розшифрування даних на зашифрованому диску GELI. Кожен диск GELI має свій власний Master Key, який створюється при створенні зашифрованого диску;

– `MD5 hash` - це результат хеш-функції, яка перетворює вихідні дані будь-якої довжини у фіксований розмірний рядок символів. Цей рядок використовується для перевірки цілісності вихідних даних, оскільки навіть невеликі зміни в початкових даних призведуть до зміни MD5 хешу.

GELI використовує алгоритм PBKDF2 для генерації Master Key з парольної фрази. PBKDF2 - це функція, що використовується для похідного ключа, що базується на паролі, тобто ключа, який може бути виведений з пароля.

PBKDF2 працює на основі HMAC, який забезпечує аутентифікацію повідомлення та цілісність. При генерації Master Key з парольної фрази, PBKDF2 використовується з сіллю (salt), що забезпечує додатковий рівень безпеки.

Після завершення роботи з зашифрованим диском, його необхідно розмонтувати. Для цього використайте команду:

```
# geli detach /dev/da5
```

За допомогою GELI та не складної послідовність дій було створено зашифрований диск з шифруванням AES 256 на операційній системі FreeBSD.

GELI є потужним механізмом шифрування на рівні блоків у FreeBSD та інших BSD-подібних операційних системах. GELI дозволяє зашифрувати практично будь-який пристрій для збереження даних та забезпечити захист від несанкціонованого доступу до даних, які зберігаються на цьому пристрої. Крім того, GELI забезпечує можливість шифрування з різними рівнями безпеки.

2.1.3 Процедура шифрування даних за допомогою EncFS

У FreeBSD EncFS може бути встановлений зі стандартних репозиторіїв. Після встановлення програми віртуальний зашифрований том може бути створений за допомогою командного рядка. Після створення тому його можна монтувати як звичайну файлову систему, що забезпечує доступ до зашифрованих файлів [9].

Послідовність шифрування за допомогою EncFS в FreeBSD з використанням шифрування з параметрами за замовчуванням наступна:

1) Створюємо теку для монтажу зашифрованого тому та теку для зберігання зашифрованої інформації (Рис. 2.6).

```
root@frebsd123srv:/# mkdir /root/encrypted
root@frebsd123srv:/# mkdir /root/decoded
```

Рисунок 2.6 – Створення тек

Тека /root/encrypted використовується для зберігання зашифрованих даних і тека /root/decoded для монтування зашифрованого тому.

2) Створюємо віртуальний зашифрований том за допомогою EncFS з використанням стандартного режиму шифрування за замовчуванням (Рис. 2.7).

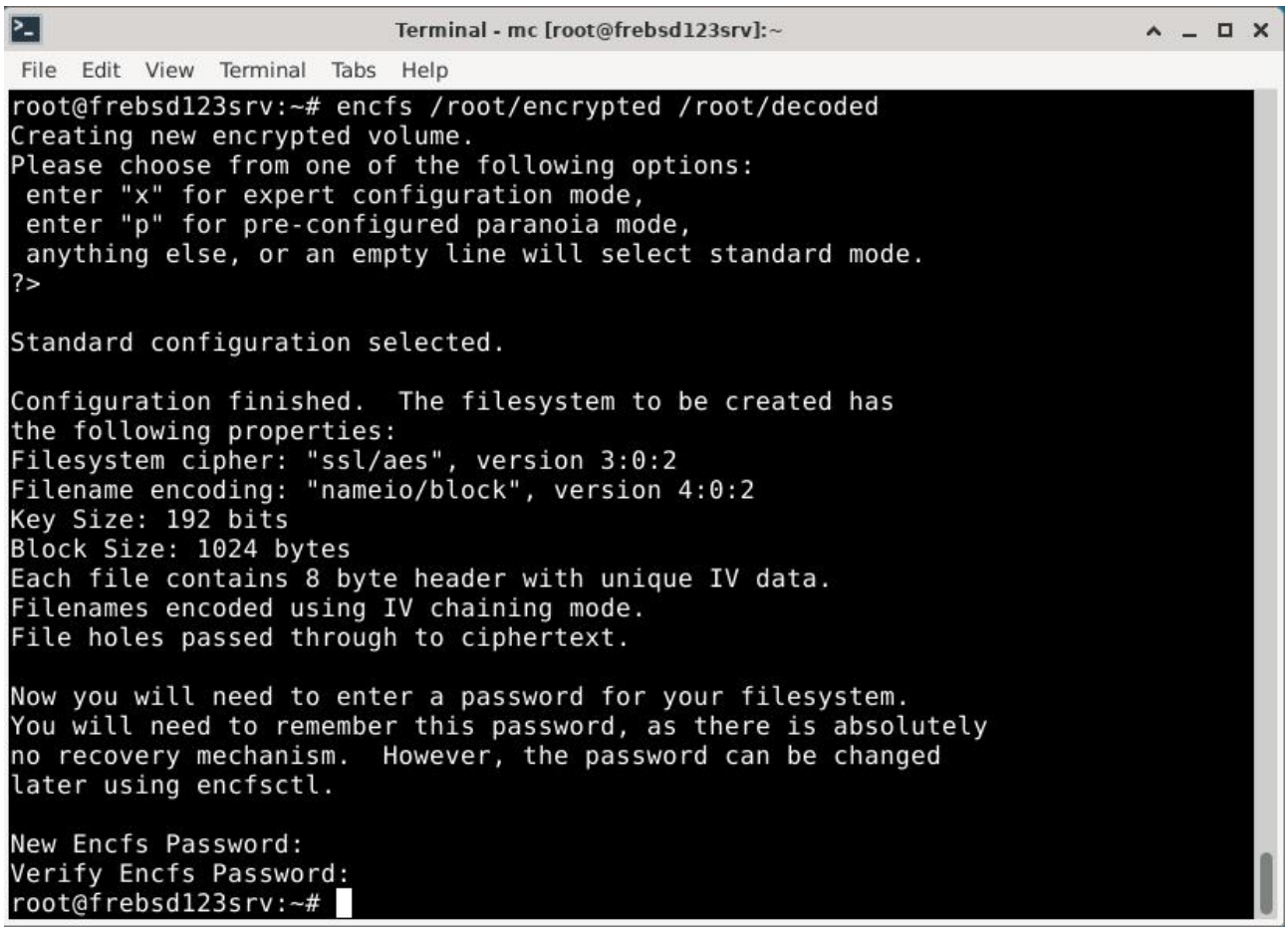

Рисунок 2.7 – Створення віртуального зашифрованого тому

Після введення команди EncFS запитає ввести пароль. Після введення пароля зашифрований том буде змонтовано у теку /root/decoded (Рис. 2.8).

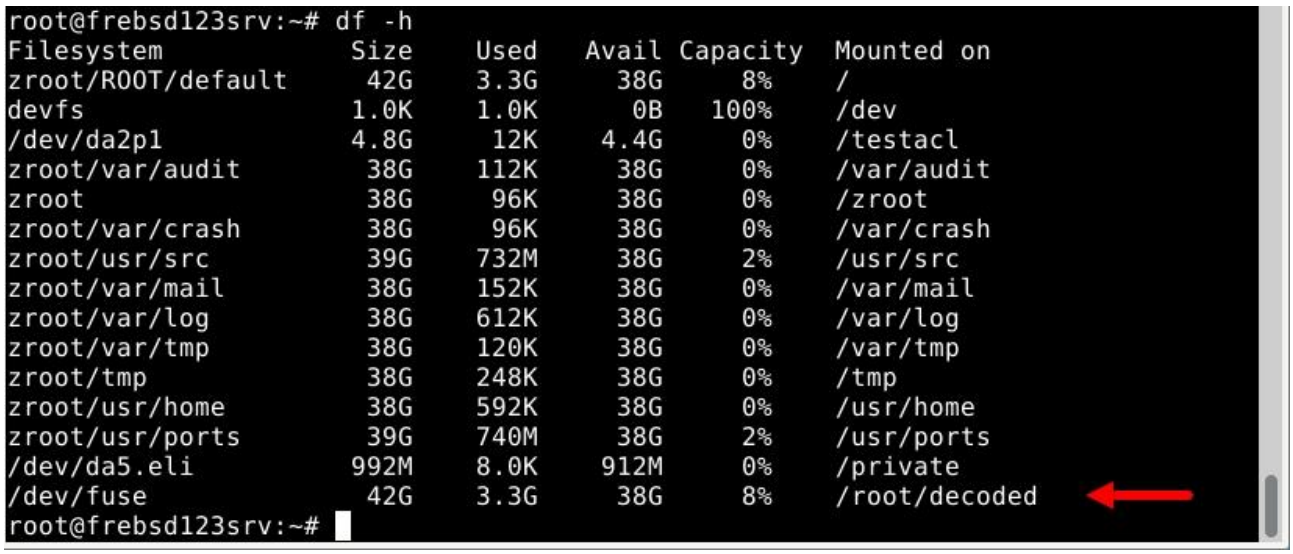

Рисунок 2.8 – Змонтований шифрований том

3) Створюємо текстовий файл testfile.txt (Рис. 2.9) з довільним вмістом (Рис. 2.10) в теці /root/decoded.

```
root@frebsd123srv:~#
root@frebsd123srv:~# cd /root/decoded/
root@frebsd123srv:~/decoded# ls -l
total 0
root@frebsd123srv:~/decoded# touch  testfile.txt
root@frebsd123srv:~/decoded# nano ./testfile.txt
```

Рисунок 2.9 – Створення текстового файлу testfile.txt

Рисунок 2.10 – Вміст текстового файлу testfile.txt

Після створення файлу та збереження його вмісту у теці /root/decoded в теці /root/encrypted буде збережено зашифрований файл (Рис 2.11 та Рис.2.12)

```
root@frebsd123srv:~/decoded#
root@frebsd123srv:~# cd /root/encrypted/
root@frebsd123srv:~/encrypted#
root@frebsd123srv:~/encrypted# ls -l
total 5
-rw-r--r--  1 root  wheel  1297 May 12 21:27 .encfs6.xml
-rw-r--r--  1 root  wheel    27 May 12 21:36 82QD5ECmltsJ5QGR3EjRXpBJ
root@frebsd123srv:~/encrypted#
```

Рисунок 2.11 – Зашифрована назва файлу testfile.txt.

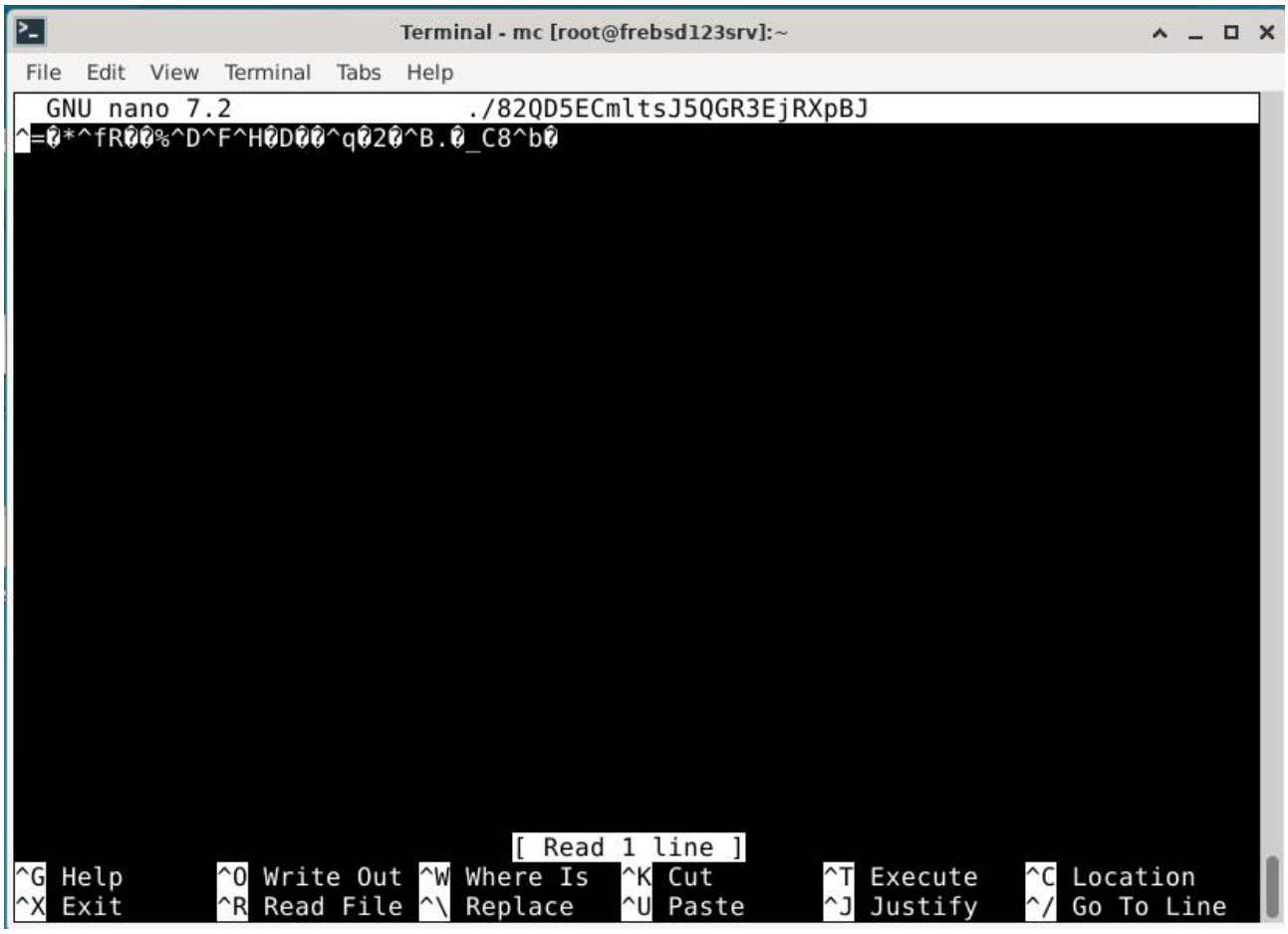

Рисунок 2.12 – Зашифрована вміст файлу testfile.txt

Ця послідовність дозволяє створювати зашифровані томи за допомогою EncFS на FreeBSD системі та забезпечити безпеку ваших конфіденційних файлів.

Управління EncFS через консоль в FreeBSD здійснюється за допомогою командного інтерфейсу `encfsctl`. Він надає користувачам можливість виконувати різні операції з зашифрованими томами. Для використання `encfsctl` необхідно мати права адміністратора або користувача, який має право на доступ до зашифрованого тому [10].

Основні операції, які можна виконати за допомогою `encfsctl`:

– `showcruft` – рекурсивний пошук по всьому тому та відображення всіх файлів які не підлягають декодуванню;

– `decode` – дозволяє вказати закодовану назву в командному рядку та відобразити розшифровану версію;

– `encode` - дозволяє вказати назву файлу в командному рядку та відобразити його закодовану версію;

– `passwd` - дозволяє змінити пароль зашифрованої файлової системи;

– `info` - виводить інформацію про зашифрований том, таку як розмір, дата створення, алгоритм шифрування тощо (Рис. 2.13).

```
root@frebsd123srv:~# encfsctl info /root/encrypted/

Version 6 configuration; created by EncFS 1.9.5 (revision 20100713)
Filesystem cipher: "ssl/aes", version 3:0:0 (using 3:0:2)
Filename encoding: "nameio/block", version 4:0:0 (using 4:0:2)
Key Size: 192 bits
Using PBKDF2, with 680355 iterations
Salt Size: 160 bits
Block Size: 1024 bytes
Each file contains 8 byte header with unique IV data.
Filenames encoded using IV chaining mode.
File holes passed through to ciphertext.

root@frebsd123srv:~#
```

Рисунок 2.13 – Вивід інформації про зашифрований том

У даному прикладі, параметри шифрування можна розшифрувати наступним чином:

– Filesystem cipher: "ssl/aes" - це алгоритм шифрування, який використовується EncFS для шифрування файлової системи. В цьому алгоритмі використовується комбінація двох криптографічних алгоритмів: SSL і AES;

– Алгоритм SSL використовується для забезпечення безпеки при передачі даних по мережі Інтернет. В EncFS він використовується для створення тунелю (тобто безпечного каналу) між файловою системою та додатком, який працює з файлами. Цей тунель захищає дані від перехоплення та зламу;

– Алгоритм AES використовується для справжнього шифрування даних на рівні файлової системи. AES є одним з найбільш захищених алгоритмів шифрування, який забезпечує високий рівень безпеки даних.

Таким чином, комбінація алгоритмів SSL та AES в "ssl/aes" забезпечує як безпеку при передачі даних між файловою системою та додатком, так і безпеку даних на рівні файлової системи.

EncFS працює в режимі шифрування CBC. У режимі CBC, перед шифруванням кожного блоку відкритого тексту, він комбінується з попереднім

зашифрованим блоком за допомогою операції XOR. Це дозволяє забезпечити високий рівень безпеки, оскільки шифрування одного блоку залежить від попереднього, що ускладнює можливість зламання шифрування.

У режимі CBC також використовується ініціалізаційний вектор (IV), який використовується для початкового блоку відкритого тексту. Ініціалізаційний вектор випадково генерується і передається разом з зашифрованим текстом, що дозволяє забезпечити унікальність шифрування і запобігти можливості аналізу шифротексту.

Після роботи з зашифрованими файлами можна відмонтувати том командою `umount /root/encrypted`.

Ця команда відмонтує зашифрований том, і файли будуть захищені шифруванням AES 256.

EncFS використовує асиметричне шифрування для захисту ключа шифрування, який використовується для шифрування та розшифрування даних. За замовчуванням EncFS використовує алгоритм шифрування RSA для асиметричного шифрування ключа. Ключ шифрування зберігається у файлі .encfs6.xml, який знаходиться у кореневій папці змонтованого зашифрованого тому (Рис. 2.14).

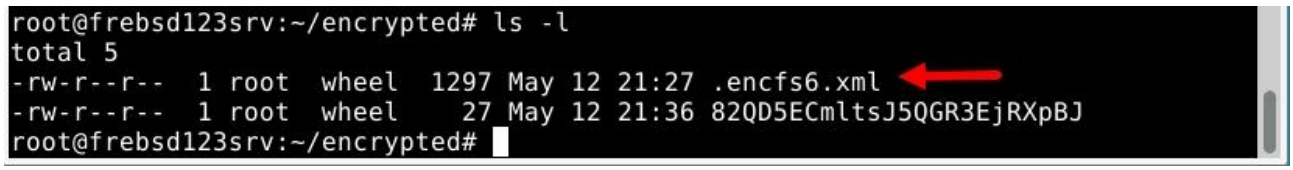

Рисунок 2.14 –Файл конфігурації EncFS

Це файл конфігурації EncFS з набором параметрів шифрування і ключів, які використовуються для захисту даних (Рис. 2.15.).

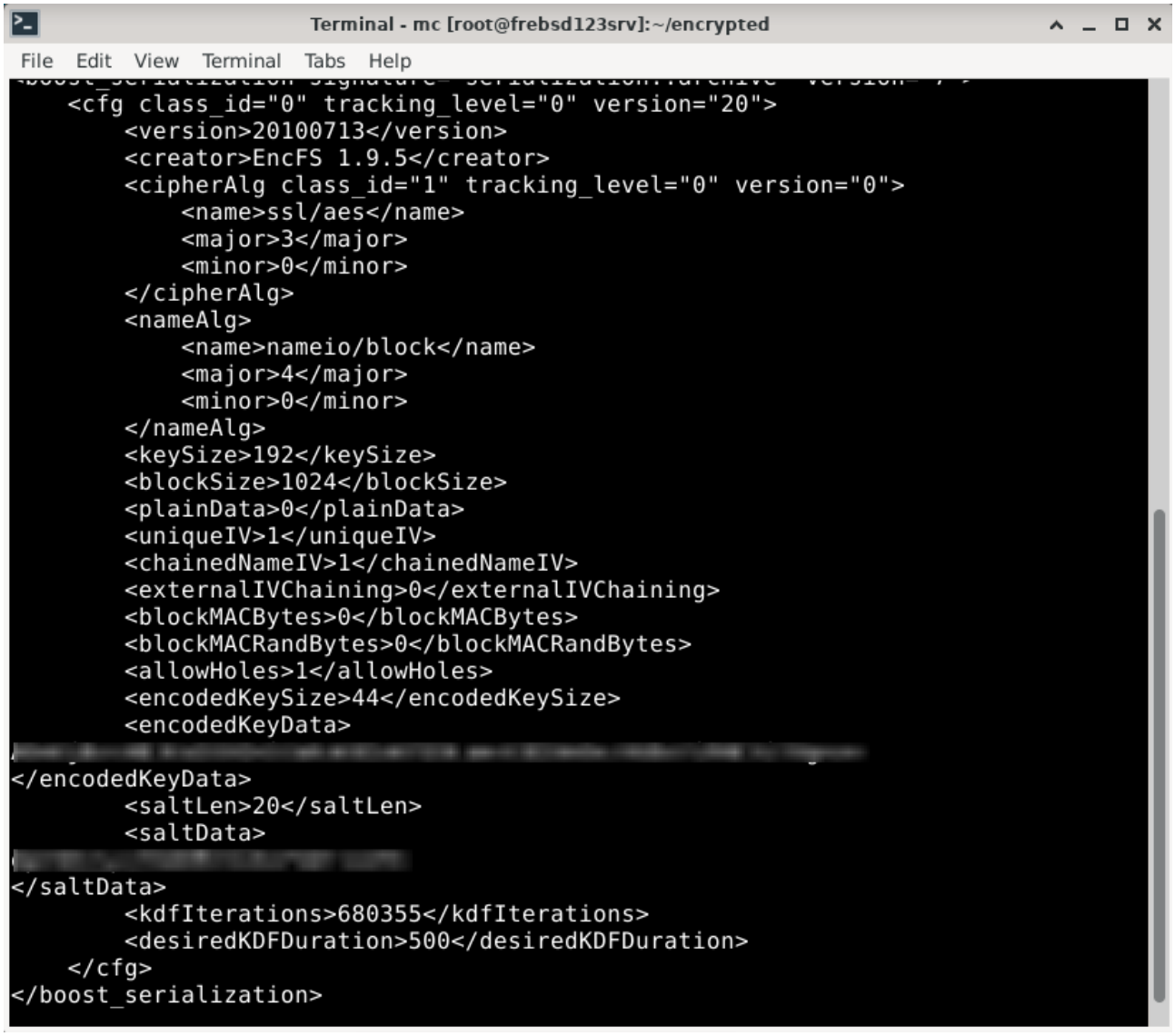

Рисунок 2.15 – Вміст файлу .encfs6.xml

Основні поля включають:

- `cipherAlg` - алгоритм шифрування, в даному випадку "ssl/aes";
- `nameAlg` - алгоритм формування імен файлів;
- `keySize` - розмір ключа шифрування в бітах, у цьому випадку 192 біта;
- `blockSize` - розмір блоку даних в байтах, у цьому випадку 1024 байт;
- `uniqueIV` - показник використання унікальних векторів ініціалізації (IV) для кожного блоку даних;
- `encodedKeyData` - ключ шифрування, закодований в base64;
- `saltData` - дані солі, також закодовані в base64;
- `kdfIterations` - кількість ітерацій для отримання ключа шифрування.

Дана не складна послідовність дій для шифрування файлів та тек забезпечує можливість створення віртуального зашифрованого тому на основі існуючої файлової системи.

Загалом, EncFS є потужним інструментом для шифрування файлових систем в FreeBSD, який дозволяє забезпечити конфіденційність та цілісність даних з високим рівнем захисту та налаштуванням під потреби користувача.

2.2 Операційна система MacOS

MacOS - це операційна система, яка розробляється компанією Apple Inc. Вона призначена для встановлення на комп'ютери Macintosh (Mac), які виробляє компанія Apple. MacOS базується на UNIX-подібній архітектурі та містить декілька унікальних функцій а також високооптимізований інтерфейс користувача. MacOS є закритою операційною системою, тобто вихідний код системи не розповсюджується публічно, і доступ до нього є обмеженим. Це дає більшу гарантію безпеки та стабільності системи, оскільки зменшує кількість потенційних вразливостей, які можуть бути використані зловмисниками.

2.2.1 Огляд можливостей шифрування в операційній системі MacOS

MacOS включає в себе ряд інструментів та функцій для безпеки та шифрування даних, які можуть допомогти зберегти конфіденційність інформації користувача.

FileVault - це вбудований в MacOS механізм шифрування даних, який дозволяє захистити файли та дані на жорстких дисках від несанкціонованого доступу.

За допомогою FileVault можна зашифрувати всі дані на жорсткому диску, включаючи файли, програми та системні дані, що забезпечує додатковий рівень безпеки для користувача.

Щоб використовувати цей інструмент, потрібно увімкнути його в налаштуваннях системи (Рис.2.16).

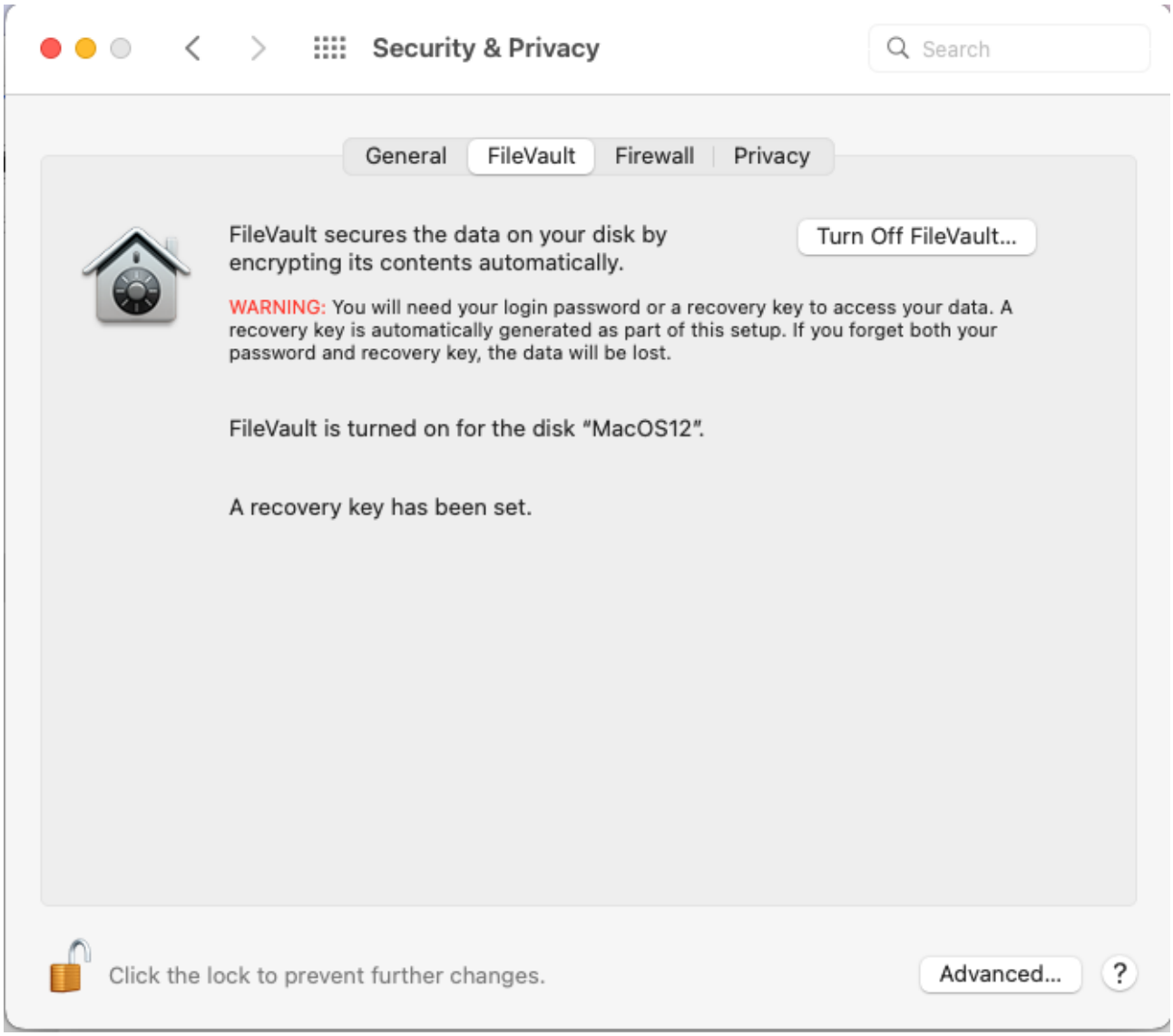

Рисунок 2.16 – Включений FileVault на системному диску

FileVault використовує сучасні криптографічні алгоритми для шифрування даних на жорсткому диску, включаючи алгоритм AES-XTS 128/256, який є одним з найбільш безпечних та ефективних алгоритмів шифрування. Цей алгоритм застосовується для шифрування даних на рівні блоків жорсткого диска, що забезпечує високий рівень захисту даних від несанкціонованого доступу [11].

FileVault дозволяє зашифрувати дані на жорсткому диску з паролем користувача, який використовується для розшифрування даних при запуску системи. Крім того, можна створити ключ відновлення, який можна використовувати для відновлення доступу до даних у випадку, якщо користувач забув свій пароль або втратив його.

За допомогою FileVault можна конвертувати існуючу файлову систему в Encrypted APFS. Apple File System Encrypted - це новітня файлова система, що підтримує шифрування даних.

Для проведення процедури шифрування можна використовувати Disk Utility.

Disk Utility - це інструмент MacOS для керування жорстким диском та створення дискових образів. Його можна використовувати для створення зашифрованого дискового образу з файловою системою Encrypted APFS, щоб зберігати конфіденційні файли (як аналог шифрування файлів або тек вибірково) або шифрування жорсткого диску або USB флеш в цілому.

FileVault є дуже ефективним засобом захисту даних на MacOS і рекомендується для використання користувачами, які працюють з конфіденційною інформацією. Додатковою перевагою FileVault є можливість використання різних методів аутентифікації, включаючи паролі, картки зі сканером відбитків пальців та Apple Watch.

Важливо також пам'ятати, що шифрування даних - це тільки один з елементів безпеки в комп'ютерній системі, і для повноцінної захисту від кібератак необхідно дотримуватися деяких правил безпеки, таких як використання сильних паролів, оновлення програмного забезпечення.

2.2.2 Шифрування жорсткого диску в цілому за допомогою Disk Utility та Encrypted APFS

На MacOS окрім шифрування системного жорстоко диску за допомогою FileVault можна також зашифрувати будь-який інший жорсткий диск або USB флеш, використовуючи вбудований інструмент Disk Utility та Encrypted APFS [12]. Encrypted APFS є доступним лише для Mac з ОС High Sierra 10.13 і новішої версії.

Щоб зашифрувати жорсткий диск, виконаємо наступні кроки:

- Відкриваємо Disk Utility, та у списку дисків зліва вибираємо жорсткий диск, який хочемо зашифрувати (Рис. 2.17).

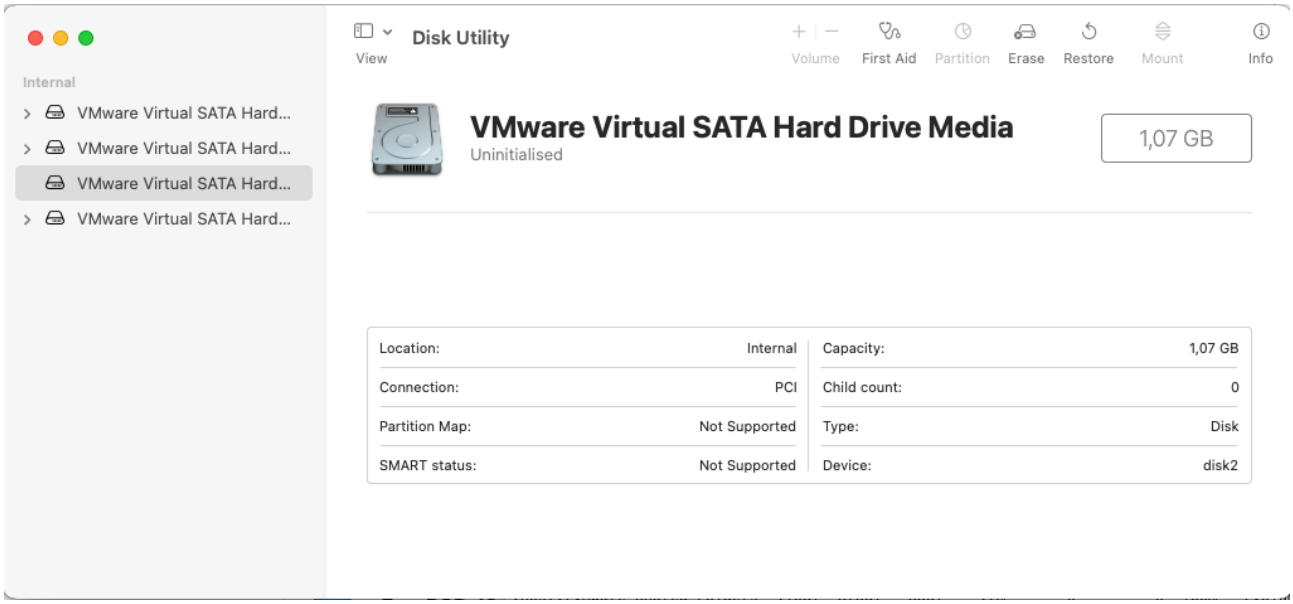

Рисунок 2.17 – Вибір диску для виконання шифрування

- Натискаємо на кнопку "Erase" вверху вікна.

Відкриється нове вікно "Erase", де вказаємо налаштування для форматування диска. Вбираємо формат "APFS Encrypted " (Рис. 2.18).

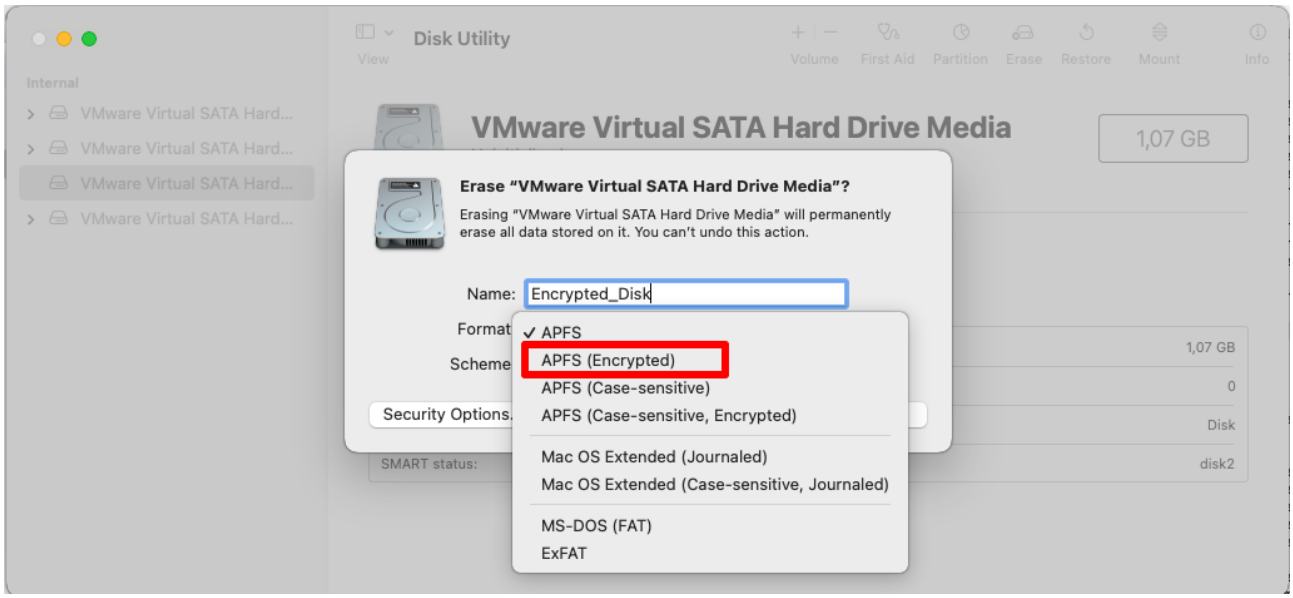

Рисунок 2.18 – Вибір шифрованої файлової системи

- Вводимо пароль, який буде використовуватися для розблокування шифрованого диска (Рис. 2.19).

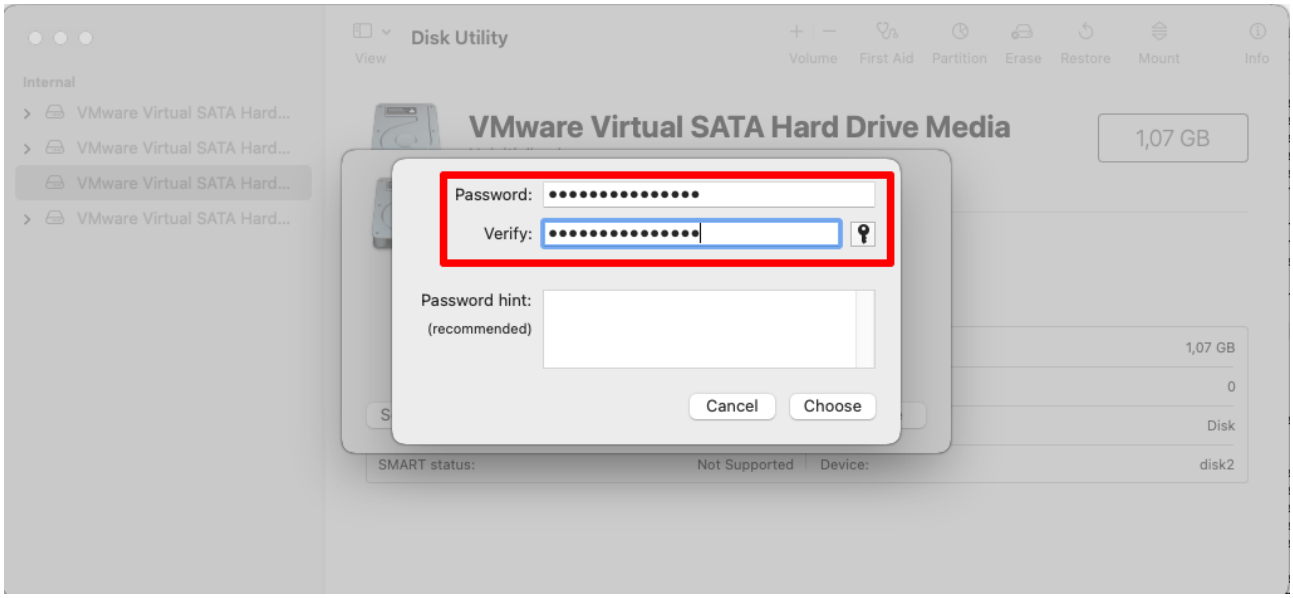

Рисунок 2.19 – Введення паролю розблокування диску

- Натискаємо "Erase" для створення та монтування зашифрованого диску (Рис. 2.20).

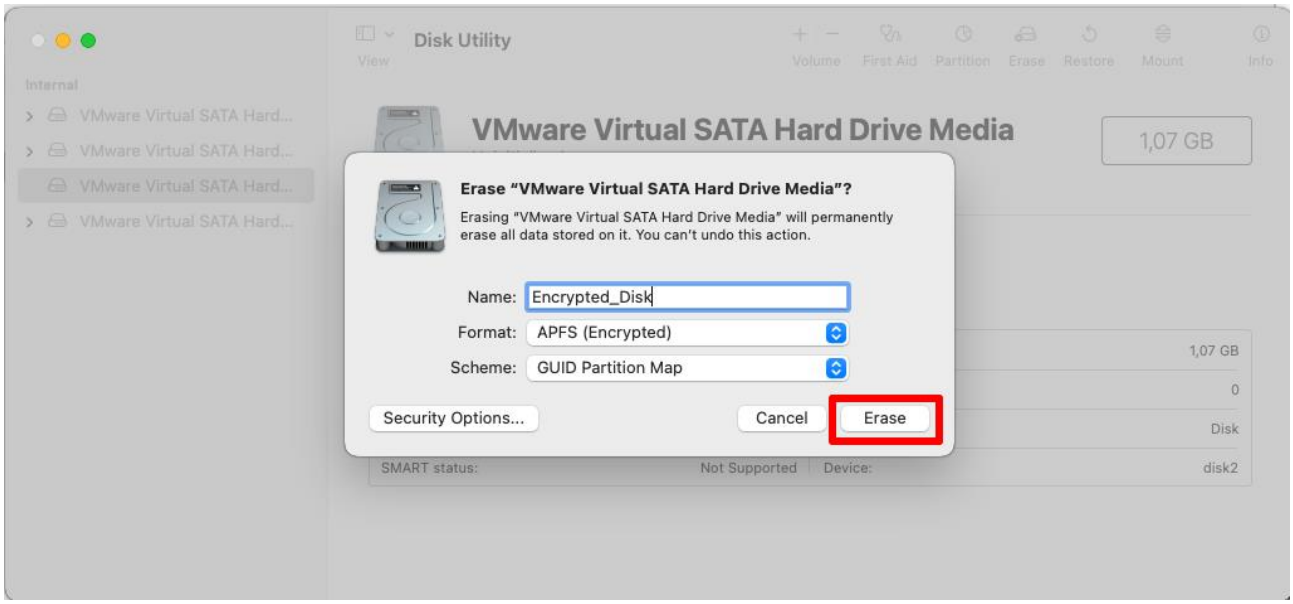

Рисунок 2.20 – Створення та монтування зашифрованого диску.

Після успішного виконання шифрування та монтування диску отримуємо повідомлення завершення (Рис 2.21). Потрібно почекати доки процес шифрування диска не буде завершено. Час, необхідний для завершення процесу, залежить від розміру диска та швидкості комп'ютера.

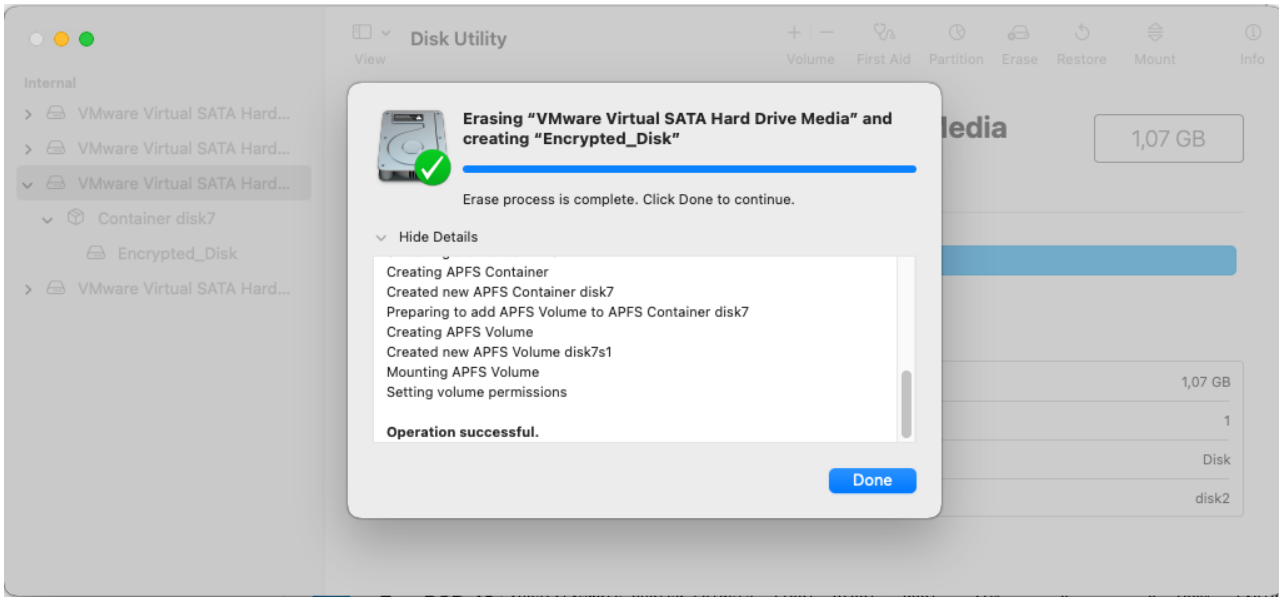

Рисунок 2.21 – Повідомлення про успішне завершення шифрування та монтування диску

На Рисунках 2.22-2.24 можна побачити що процедура шифрування пройшов успішно та диск змонтовано.

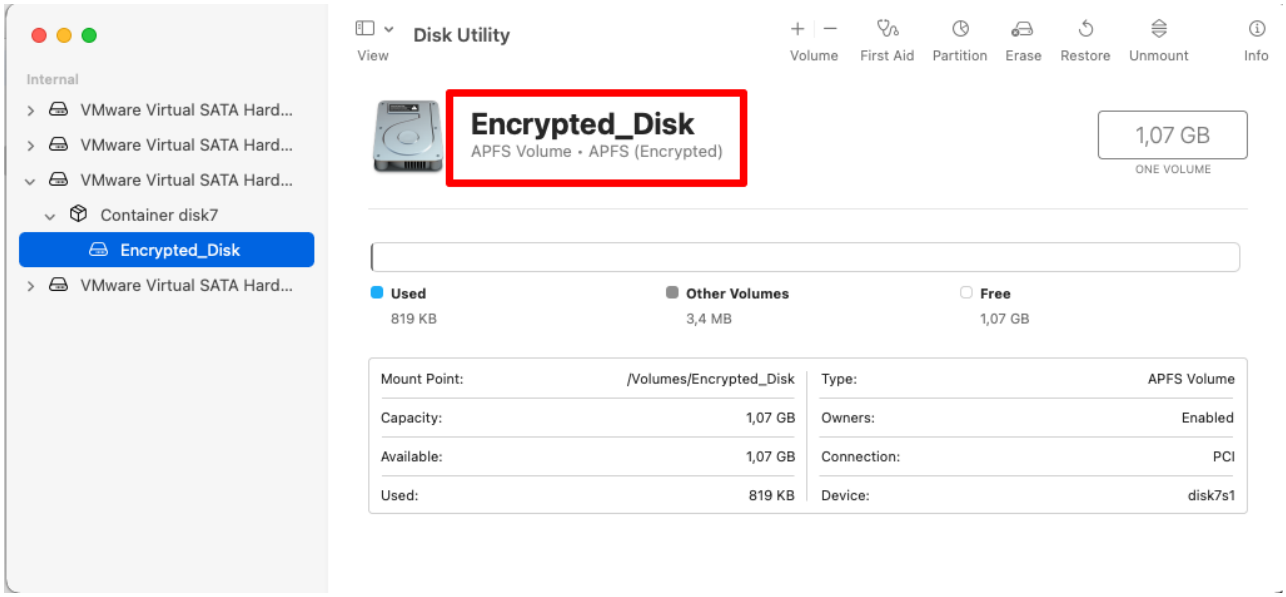

Рисунок 2.22 – Значок шифрованого диску в Disk Utility

```
macos1251ws:~ root# df -h
Filesystem                    Size   Used  Avail Capacity iused      ifree %iused  Mounted on
/dev/disk5s5s1                80Gi   22Gi   20Gi    52%  502048  213443120    0%   /
devfs                        199Ki  199Ki    0Bi   100%     688          0  100%   /dev
/dev/disk5s4                  80Gi  1.0Mi   20Gi     1%       1  213443120    0%   /System/Volumes/VM
/dev/disk5s2                  80Gi  284Mi   20Gi     2%    1071  213443120    0%   /System/Volumes/Preboot
/dev/disk5s6                  80Gi  102Mi   20Gi     1%     402  213443120    0%   /System/Volumes/Update
/dev/disk5s1                  80Gi   36Gi   20Gi    64%  955476  213443120    0%   /System/Volumes/Data
/dev/disk5s5                  80Gi   22Gi   20Gi    52%  502164  213443120    0%   /System/Volumes/Update/mnt1
map auto_home                  0Bi    0Bi    0Bi   100%       0          0  100%   /System/Volumes/Data/home
.host:/VMware Shared Folders  910Gi  875Gi   35Gi    97%       0          0  100%   /Volumes/VMware Shared Folders
/dev/disk3s1                  1.0Gi  812Ki  1.0Gi     1%      95   10444480    0%   /Volumes/Untitled
/dev/disk6s1                  100Gi   48Gi   51Gi    49%     222  539828040    0%   /Volumes/HDD2MacOS
/dev/disk7s1                  1.0Gi  800Ki  1.0Gi     1%      92   10444520    0%   /Volumes/Encrypted_Disk
macos1251ws:~ root#
```

Рисунок 2.23 – Вивід команди df –h для підтвердження монтування

Рисунок 2.24 – Вивід інформації про розділи зашифрованого диску

Після завершення процесу шифрування диска при кожному запуску вашого комп'ютера, вам потрібно буде вводити пароль для розблокування диска.

2.2.3 Створення зашифрованого дискового образу за допомогою Disk Utility з файловою системою Encrypted APFS

Створення зашифрованого дискового образу за допомогою Disk Utility з файловою системою Encrypted APFS - це процес створення віртуального дискового образу, який буде зашифрований з використанням файлової системи APFS. Disk Utility є утилітою на macOS, яка дозволяє керувати дисками, образами дисків і файловими системами.

Зашифрований образ диска з файловою системою Encrypted APFS дозволяє захистити дані, шифруючи їх та забезпечуючи доступ лише з використанням правильного пароля або ключа шифрування.

Для створення зашифрованого дискового образу з файловою системою Encrypted APFS за допомогою Disk Utility на macOS, виконаємо наступні кроки:

- Відкриваємо програму Disk Utility. Виберіть меню "File" та вибираємо "Blank image" (Рис.2.25);

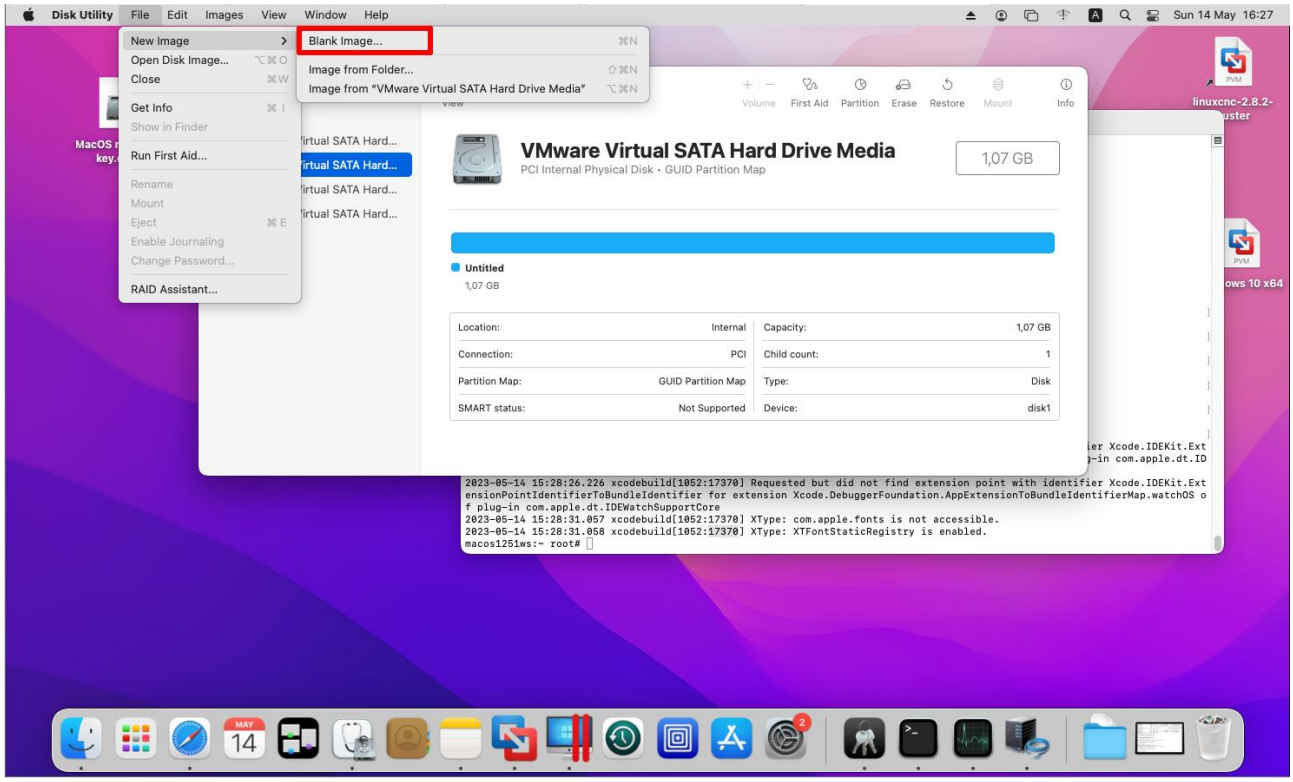

Рисунок 2.25 – Вибір методу створення дискового образу

- Вибираємо назву дискового образу, місце зберігання, шифрування та інші параметри (Рис. 2.26);

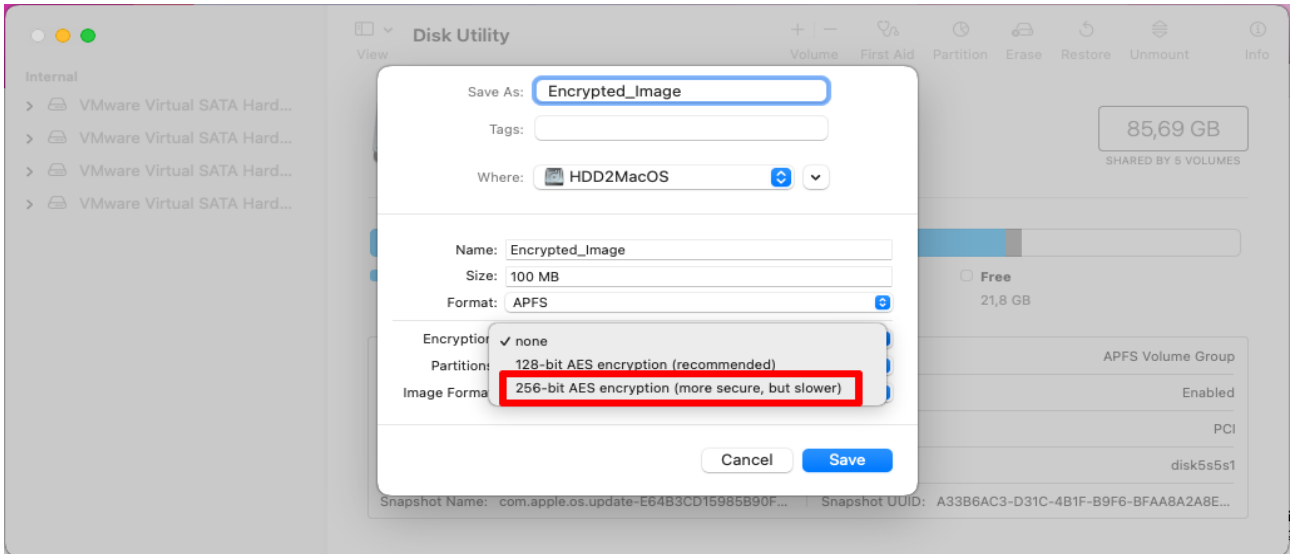

Рисунок 2.26 – Вибір параметрів та характеристик дискового образу

- Вводимо пароль, який буде використовуватися для розблокування дискового образу (Рис. 2.27).

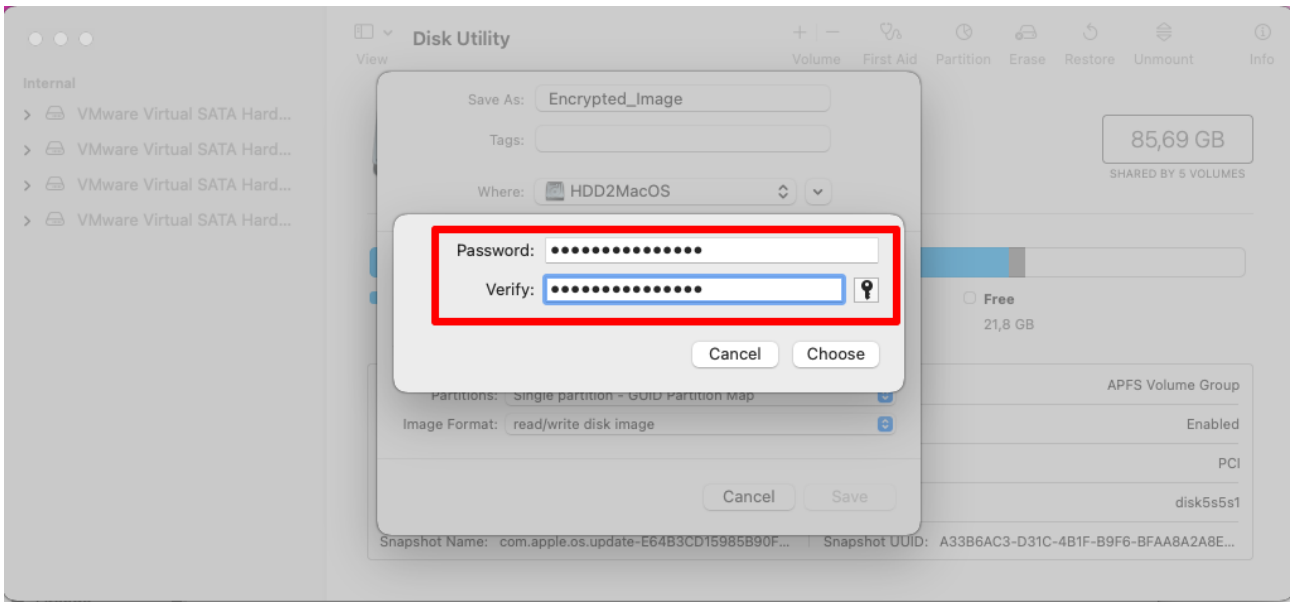

Рисунок 2.27 – Ввід пароля для розблокування дискового образу

- Створюємо дисковий образ (Рис. 2.28).

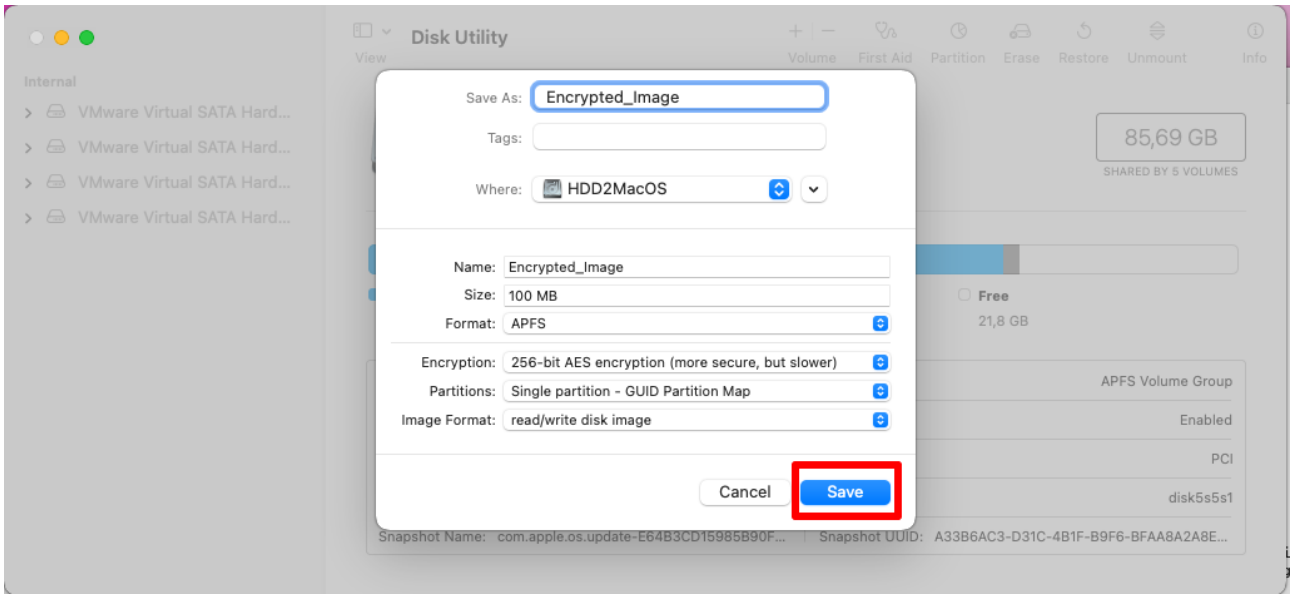

Рисунок 2.28 – Створення дискового образу

- Натискаємо на кнопку "Erase" вверху вікна.

Відкриється нове вікно "Erase" (Рис.2.29), де вказуємо налаштування для форматування диска. Вбираємо формат "APFS Encrypted " (Рис. 2.30).

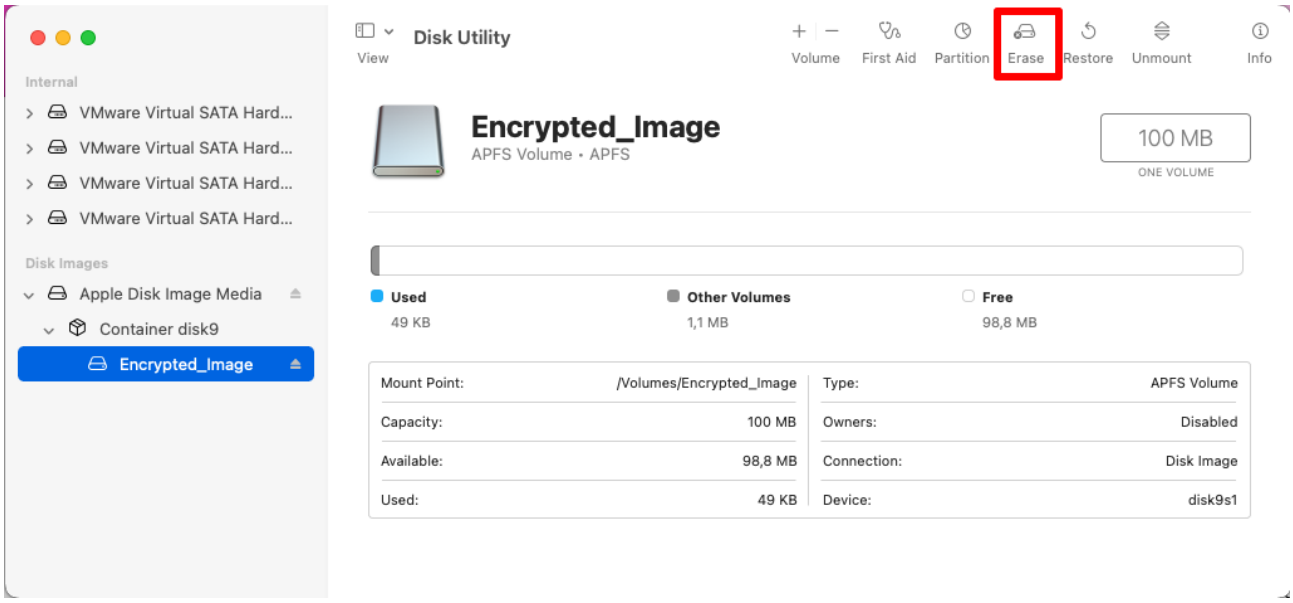

Рисунок 2.29 – Запуск послідовності створення файлової системи APFS Encrypted

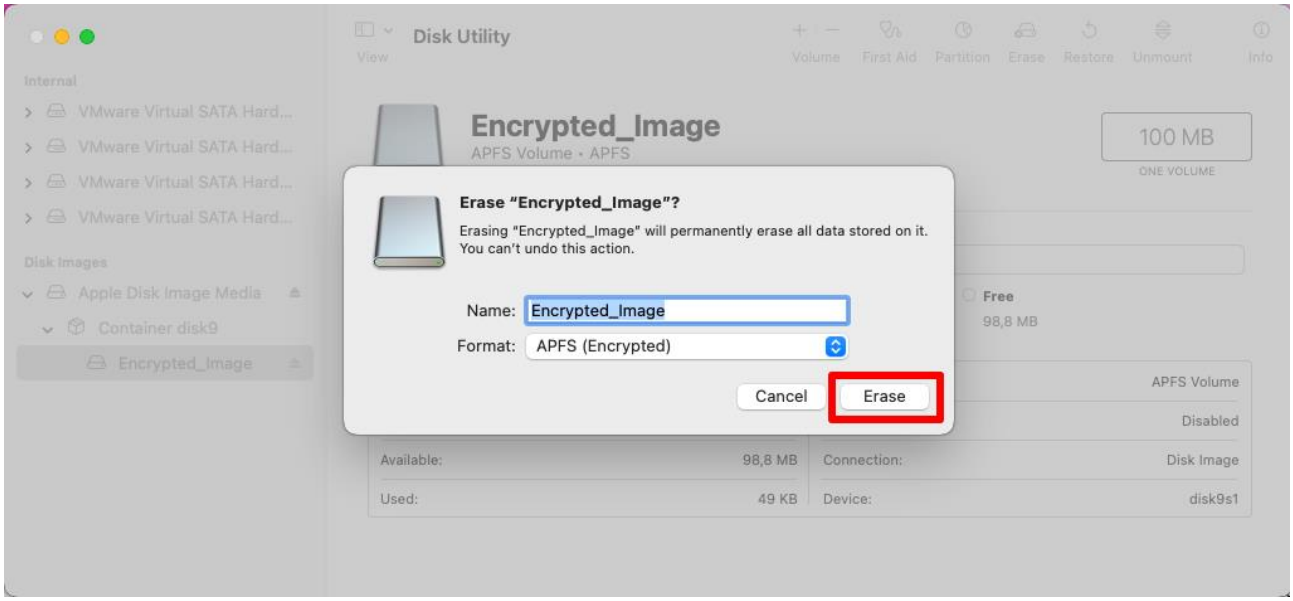

Рисунок 2.30 – Вибір шифрованої файлової системи

- Вводимо пароль, який буде використовуватися для розблокування шифрованого диска (Рис. 2.31).

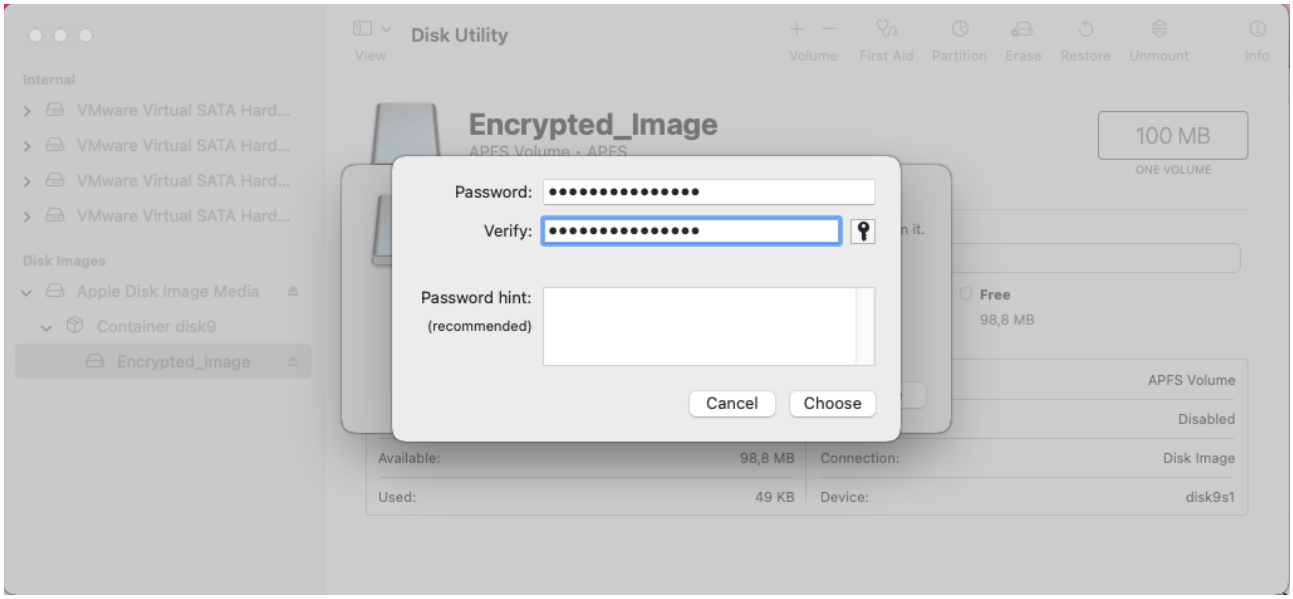

Рисунок 2.31 – Введення паролю розблокування диску

На рисунку 2.32 можна побачити що процедура шифрування пройшла успішно та диск змонтовано.

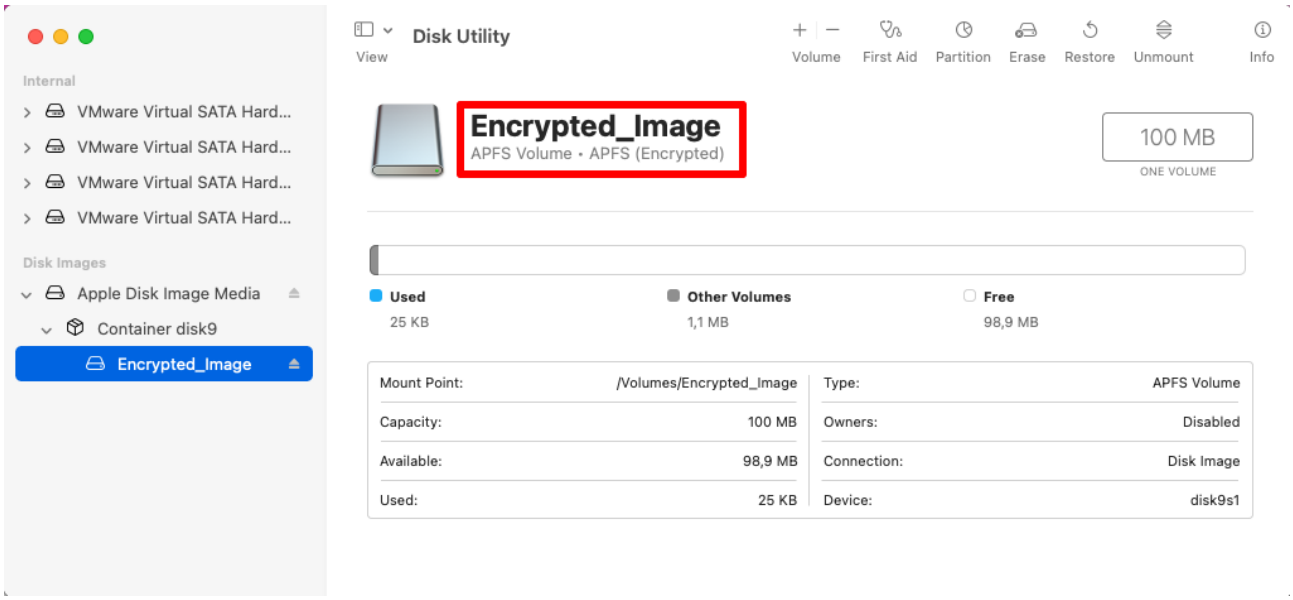

Рисунок 2.32 – Значок шифрованого диску в Disk Utility

Для перевірки чи коректно працює дисковий образ створимо текстовий файл під назва APFStestfile.txt (Рис. 2.33) з довільним текстом (Рис.2.34).

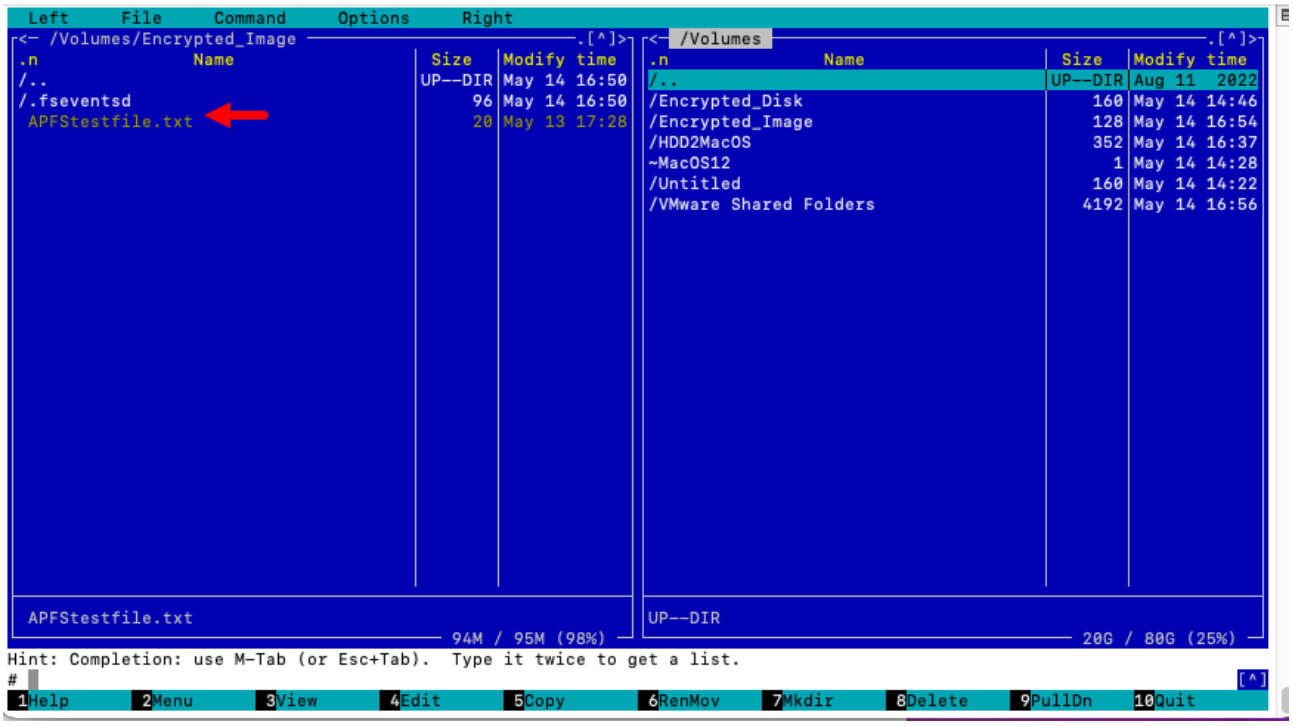

Рисунок 2.33 – Текстовий файл APFStestfile.txt

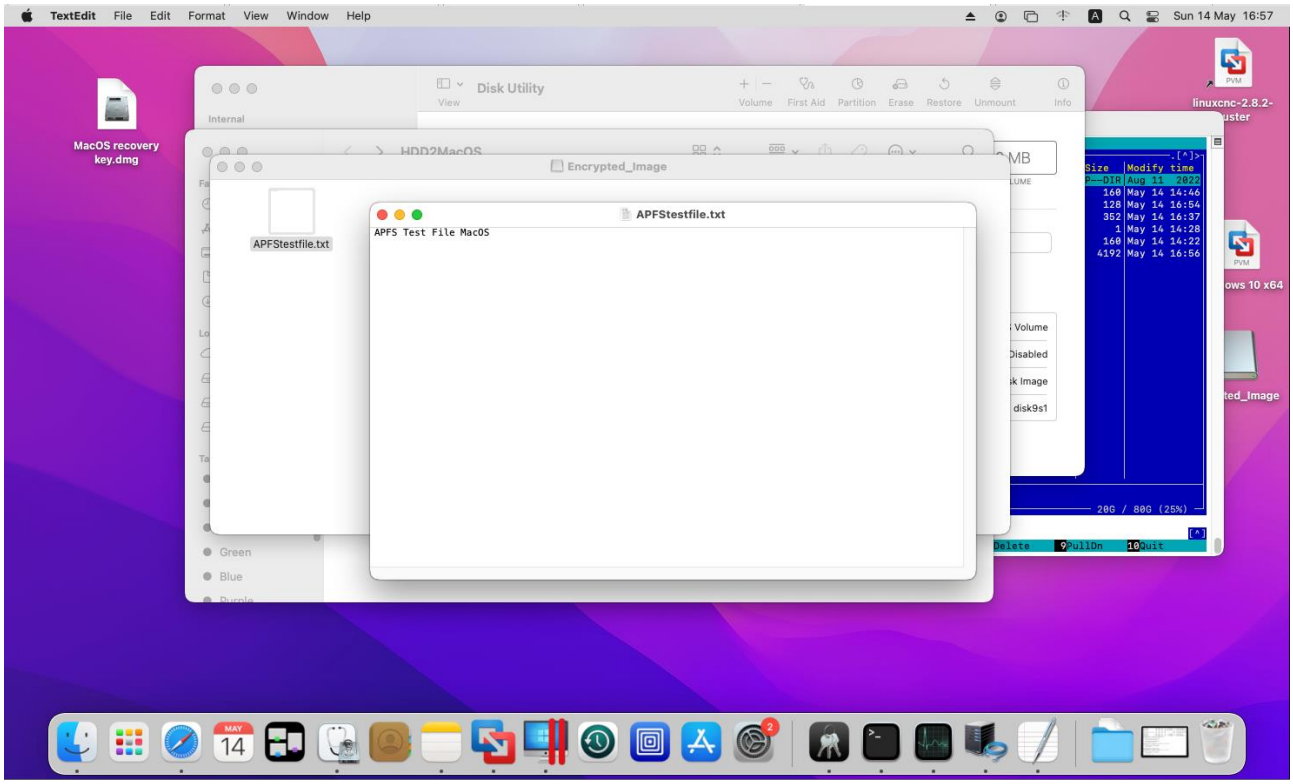

Рисунок 2.34 – Вміст текстового файлу APFStestfile.txt

Вайл APFStestfile.txt створився та є можливість його редагування без виникнення помилок. Отже шифрований образ диску Encrypted_Image (Рис.2.35) працює коректно.

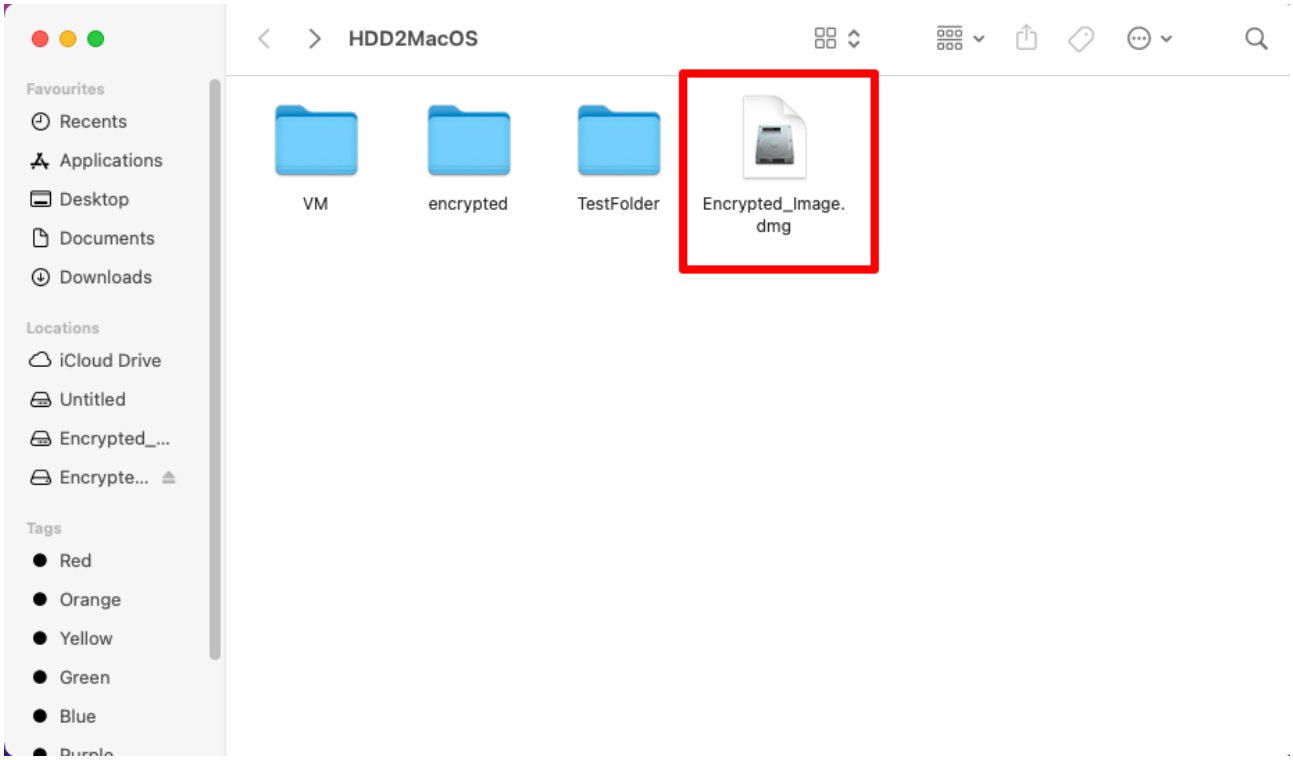

Рисунок 2.35 – Образ диску Encrypted_Image

Доступ до вмісту образу диску можливий лише після введення пароля для розблокування та монтування диску. Після цього його можна використовувати як будь-який інший диск з файловою системою APFS.

Файлова система APFS Encrypted є потужним засобом захисту даних на macOS. Вона використовує сучасні алгоритми шифрування, такі як AES-XTS з ключем довжиною 256 біт, які забезпечують високий рівень безпеки і захисту даних від несанкціонованого доступу.

3 СУМІСНІСТЬ МЕТОДІВ ШИФРУВАННЯ, ПЕРЕВАГИ ТА НЕДОЛІКИ

3.1 Переваги та недоліки GELI та EncFS в FreeBSD

GELI та EncFS - це два різні методи шифрування, які можуть використовуватися в операційній системі FreeBSD. Обидва методи мають свої переваги та недоліки, які слід враховувати перед їх використанням.

Переваги GELI*:*

- Вбудована підтримка. GELI вбудований в ядро FreeBSD, що означає, що його можна використовувати без необхідності встановлення додаткових програм або драйверів.
- Розширені можливості шифрування. GELI підтримує різні алгоритми шифрування, включаючи AES-XTS, AES-CBC, Blowfish-CBC, Camellia-CBC, 3DES-CBC.
- Гнучкі налаштування. GELI надає гнучкі налаштування для шифрування, включаючи можливість встановлювати пароль на ключі шифрування або використовувати ключі на основі файлів, що забезпечує високий рівень безпеки.
- Проста утиліта керування. Утиліта GELI дозволяє створювати, налаштовувати та управляти зашифрованими томами даних. Це робить її дуже зручним для користувачів.

Основні недоліки GELI*:*

- Потреба у вводі пароля під час завантаження системи. Це може бути не зручно в разі віддаленого доступу або автоматичного перезавантаження системи після відключення електропостачання.
- Немає підтримки динамічного збільшення розміру зашифрованого об'єму даних. Розмір зашифрованого об'єму даних встановлюється при створенні тому і не може бути змінений динамічно простим методом.

- Потенційна вразливість при атаках з використанням збоїв пам'яті. Існує можливість отримання доступу до зашифрованих даних, якщо зловмисник може отримати доступ до пам'яті, де зберігається ключ шифрування.
- Відсутність можливості шифрування окремих файлів. GELI шифрує всю файлову систему, тому немає можливості зашифрувати окремі файли та теки.

Переваги EncFS*:*

- Простота використання. EncFS є дуже простим у використанні. Його можна швидко налаштувати за допомогою командного рядка.
- Гнучкі налаштування. EncFS надає гнучкі налаштування для шифрування, включаючи можливість вибирати алгоритми шифрування, режими роботи та ключі шифрування.
- Крос-платформність. EncFS підтримується на різних операційних системах, таких як Linux та Windows, що дозволяє використовувати його на різних платформах.

Основні недоліки EncFS*:*

- Не є вбудлваним в FreeBSD. EncFS є додатковим компонентом, який потрібно встановлювати окремо. Це може створювати додаткові проблеми зі сумісністю та безпекою.
- Недостатня швидкодія. Через складність процесу шифрування та дешифрування, EncFS може працювати довше.
- Проблеми з сумісністю версій. Якщо файли EncFS будуть перенесені на систему з іншою версією EncFS або на іншу платформу, можуть виникнути проблеми з доступом до файлів або їх розшифруванням.
- Потенційні проблеми зі сумісністю файлових систем. EncFS не завжди добре працює зі всіма файловими системами. Це може призвести до втрати даних або інших проблем з файлами.

- Обмежена функціональність: EncFS не має всіх функцій, які можуть бути необхідні для користувачів. Наприклад, він не надає можливості створювати бекапи або робити снапшоти файлової системи.

Загалом GELI є потужним інструментом з глибокою інтеграцією в операційну систему FreeBSD, в той час як EncFS є простим у використанні та більш портативним. Вибір між ними залежить від конкретних потреб, рівня безпеки та продуктивності, які ви бажаєте отримати. Для прийняття найкращого рішення щодо шифрування даних в FreeBSD потрібно оцінити вимоги сценарію використання.

3.2 Переваги та недоліки Encrypted APFS в MacOS

Encrypted APFS є файловою системою з вбудованою підтримкою шифрування в операційній системі MacOS. Ця технологія надає додатковий рівень безпеки для зберігання і захисту даних.

Основні переваги Encrypted APFS:

- Висока швидкодія. Шифрування та розшифрування в Encrypted APFS відбувається досить швидко і без зайвих затримок.
- Інтеграція зі системою. Encrypted APFS є стандартним методом шифрування для MacOS, що забезпечує відмінну інтеграцію зі всіма компонентами операційної системи.
- Висока безпека. Encrypted APFS використовує потужні алгоритми шифрування, такі як AES, для захисту інформації.
- Простота використання. Шифрування та розшифрування з Encrypted APFS досить прості та зрозумілі для користувача.

Недоліки Encrypted APFS:

- Обмежені можливості. Encrypted APFS працює тільки на пристроях MacOS, що виключає його використання на пристроях з іншими операційними системами, такими як Windows, Linux або FreeBSD.
- Потреба високої продуктивності. Шифрування та розшифрування даних на Encrypted APFS можуть впливати на продуктивність пристрою. Крім того, деякі додатки можуть працювати повільніше на зашифрованих дисках.
- Обмеженість у використанні на старіших пристроях. Encrypted APFS підтримується на MacOS з версії High Sierra і вище, що обмежує його використання на старіших пристроях.

Encrypted APFS забезпечує надійне шифрування даних в MacOS. Вона забезпечує високий рівень безпеки та захисту від несанкціонованого доступу до даних. Проте, вона має обмежену сумісність з іншими операційними системами.

3.3 Сумісність методів шифрування

Сумісність методів шифрування - це можливість обміну зашифрованими даними між системами FreeBSD та MacOS. Як вже було сказано кожна з цих систем має свої власні надійні методи шифрування і формати файлових систем, але вони несумісними один з одними без використання додаткового програмного забезпечення. Коректна робота даного програмного забезпечення не гарантується і як наслідок це може привести до втрати даних. Єдиним надійним та безпечним методом розшифрування даних є використання віртуалізації.

За допомогою програм, таких як VirtualBox, VMware Fusion або Parallels Desktop, можна створити віртуальну машину з FreeBSD на Mac. Це дозволить працювати з GELI та EncFS в FreeBSD безпосередньо на MacOS через віртуальну машину. Це також дасть можливість розшифрувати дані в віртуальній машині FreeBSD та відкрити потрібні файли через SFTP на MacOS.

Проведемо розшифрування тестового USB флеш диску (Рис.3.1) зашифрованого в FreeBSD за допомогою GELI. Основні моменти шифрування USB флеш в FreeBSD показано на рисунках 3.2-3.4.

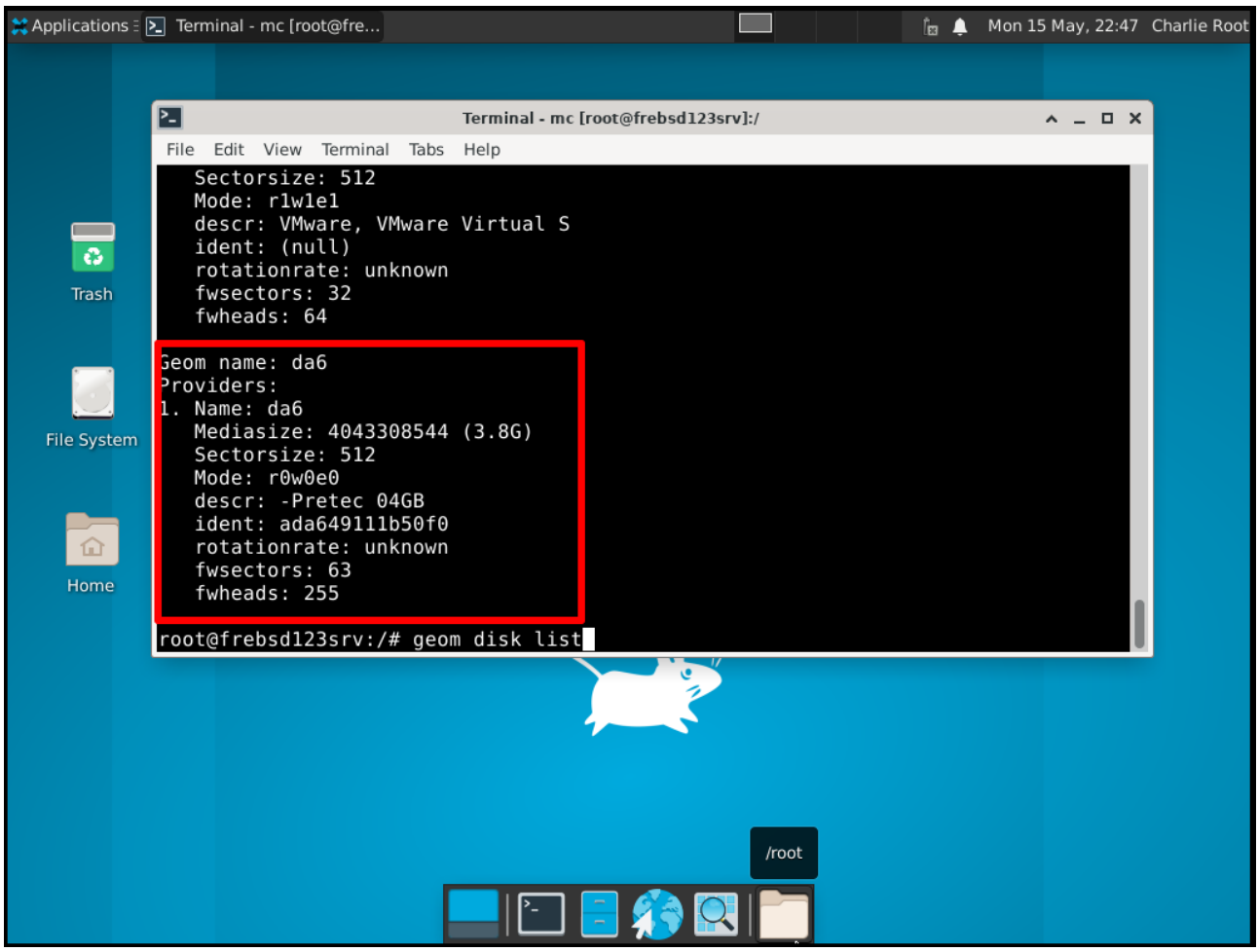

Рисунок 3.1 – Тестовий USB флеш

```
root@frebsd123srv:~/encrypted# geli init -b -s 4096 -e aes -l 256 /dev/da6
Enter new passphrase:
Reenter new passphrase:

Metadata backup for provider /dev/da6 can be found in /var/backups/da6.eli
and can be restored with the following command:

        # geli restore /var/backups/da6.eli /dev/da6

root@frebsd123srv:~/encrypted#
```

Рисунок 3.2– Шифрування USB флеш за допомогою GELI

```
root@frebsd123srv:/# geli attach /dev/da6
Enter passphrase:
root@frebsd123srv:/# ls -l /dev/da6*
crw-r-----  1 root  operator  0x8a May 15 22:51 /dev/da6
crw-r-----  1 root  operator  0x8b May 15 22:52 /dev/da6.eli
root@frebsd123srv:/# newfs /dev/da6.eli
/dev/da6.eli: 3856.0MB (7897080 sectors) block size 32768, fragment size 4096
        using 7 cylinder groups of 626.09MB, 20035 blks, 80256 inodes.
super-block backups (for fsck_ffs -b #) at:
 192, 1282432, 2564672, 3846912, 5129152, 6411392, 7693632
root@frebsd123srv:/# mount /dev/da6.eli /encrypted_flash
root@frebsd123srv:/#
```

Рисунок 3.3 – Створення файлової системи та монтування USB флеш в FreeBSD

```
root@frebsd123srv:/# df -h
Filesystem            Size    Used   Avail Capacity  Mounted on
zroot/ROOT/default     42G    3.3G     38G     8%    /
devfs                 1.0K    1.0K      0B   100%    /dev
/dev/da2p1            4.8G     12K    4.4G     0%    /testacl
zroot/var/audit        38G    112K     38G     0%    /var/audit
zroot                  38G     96K     38G     0%    /zroot
zroot/var/crash        38G     96K     38G     0%    /var/crash
zroot/usr/src          39G    732M     38G     2%    /usr/src
zroot/var/mail         38G    156K     38G     0%    /var/mail
zroot/var/log          38G    632K     38G     0%    /var/log
zroot/var/tmp          38G    120K     38G     0%    /var/tmp
zroot/tmp              38G    228K     38G     0%    /tmp
zroot/usr/home         38G    592K     38G     0%    /usr/home
zroot/usr/ports        39G    740M     38G     2%    /usr/ports
/dev/da5.eli          992M    8.0K    912M     0%    /private
/dev/fuse              42G    3.3G     38G     8%    /root/decoded
/dev/da6.eli          3.6G    8.0K    3.3G     0%    /encrypted_flash
root@frebsd123srv:/#
```

Рисунок 3.4 – Змонтована USB флеш в FreeBSD

Скопіюємо на USB флеш теку /root/encrypted (Рис.3.5) створену в пункті 2.2.2 з файлом testfile.txt. Дана тека та її вміст зашифровані за допомого EncFS.

```
root@frebsd123srv:/encrypted_flash# cd ./encrypted/
root@frebsd123srv:/encrypted_flash/encrypted# ls -l
total 8
-rw-r--r--  1 root  wheel  1297 May 12 21:27 .encfs6.xml
-rw-r--r--  1 root  wheel    27 May 12 21:36 82QD5ECmltsJ5QGR3EjRXpBJ
root@frebsd123srv:/encrypted_flash/encrypted#
```

Рисунок 3.5 – Тека encrypted з зашифрованим файлом

За допомогою гіпервізора VMware Fusion [13] створюємо віртуальну машину з FreeBSD на Mac та проведемо розшифрування вмісту USB флеш.

Це дозволить працювати з GELI та EncFS в FreeBSD безпосередньо на MacOS через віртуальну машину. Процес розшифрування показано на рисунках 3.6-3.11.

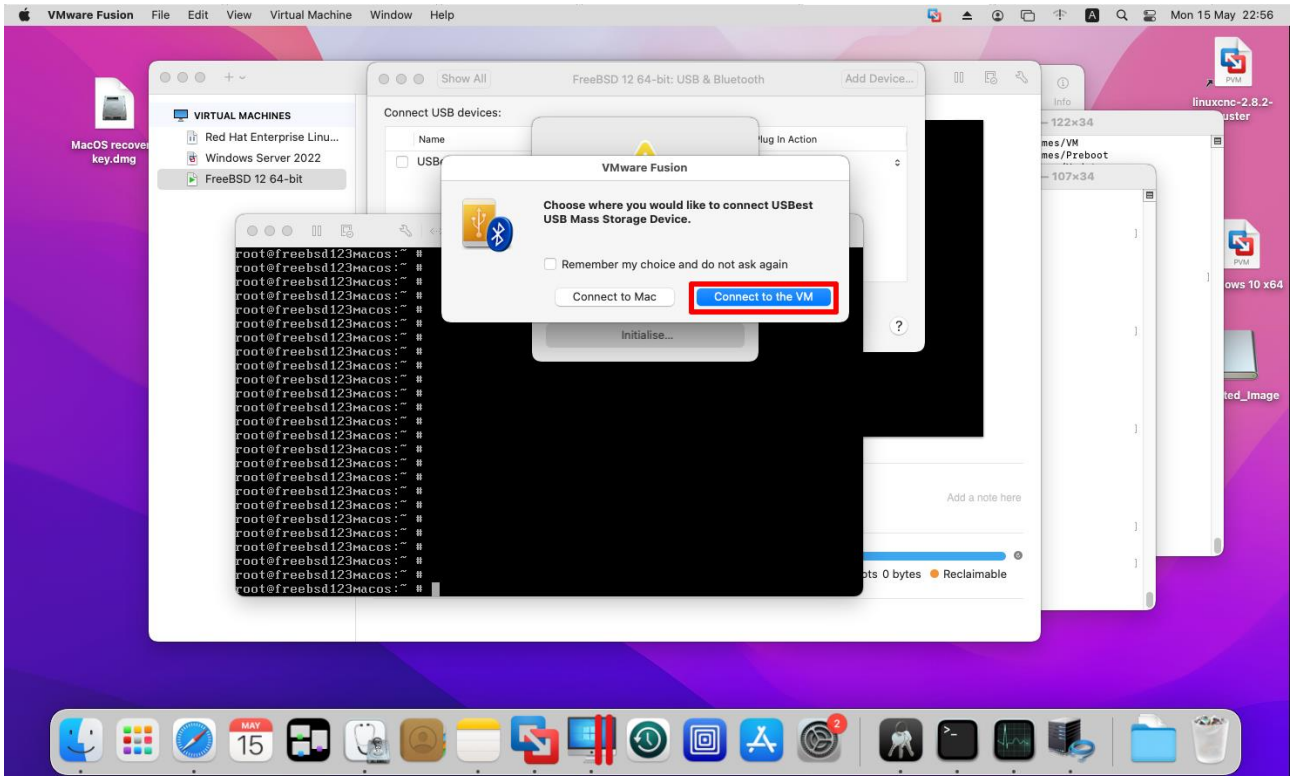

Рисунок 3.6 – Підключення USB флеш до віртуальної машини FreeBSD в MacOS

Рисунок 3.7 – Ініціалізація USB флеш в віртуальній машини FreeBSD

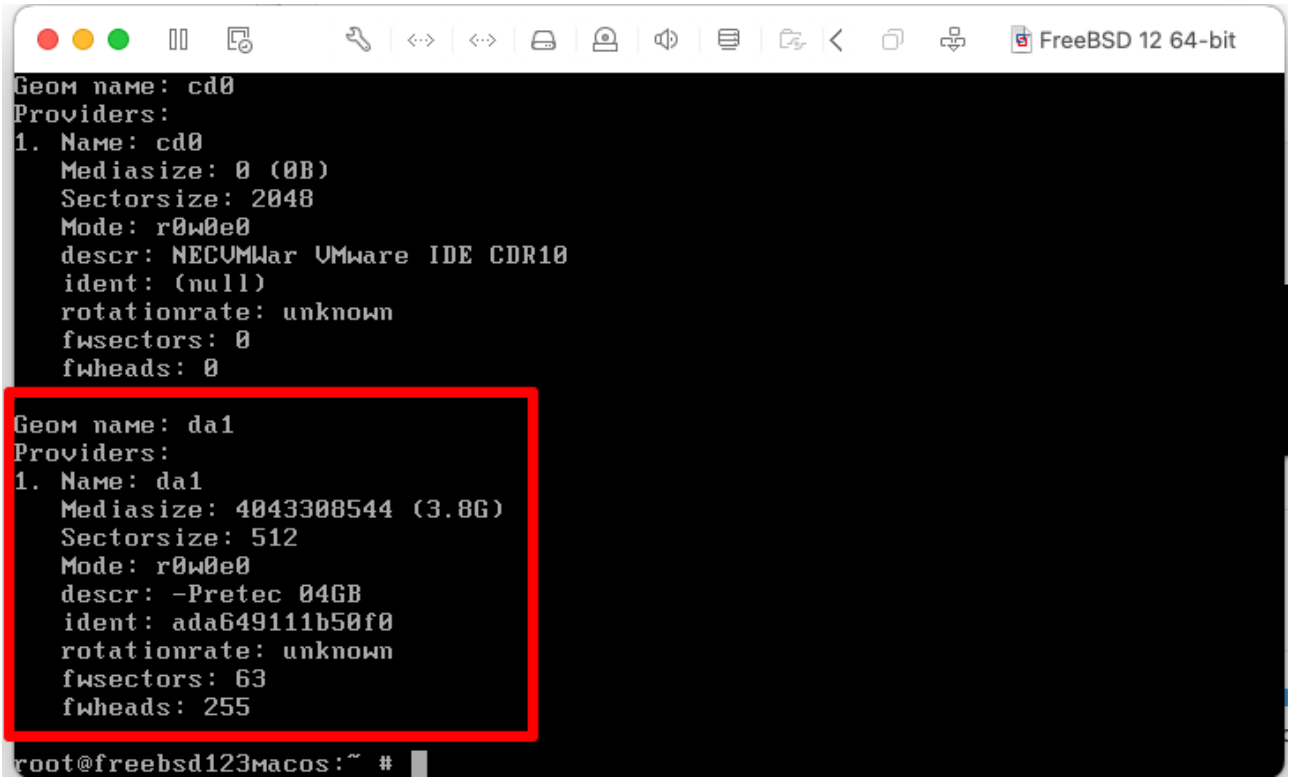

Рисунок 3.8 – Тестовий USB флеш в віртуальній машини FreeBSD

Рисунок 3.9 – Розшифрування та приєднання USB флеш в віртуальній машини FreeBSD

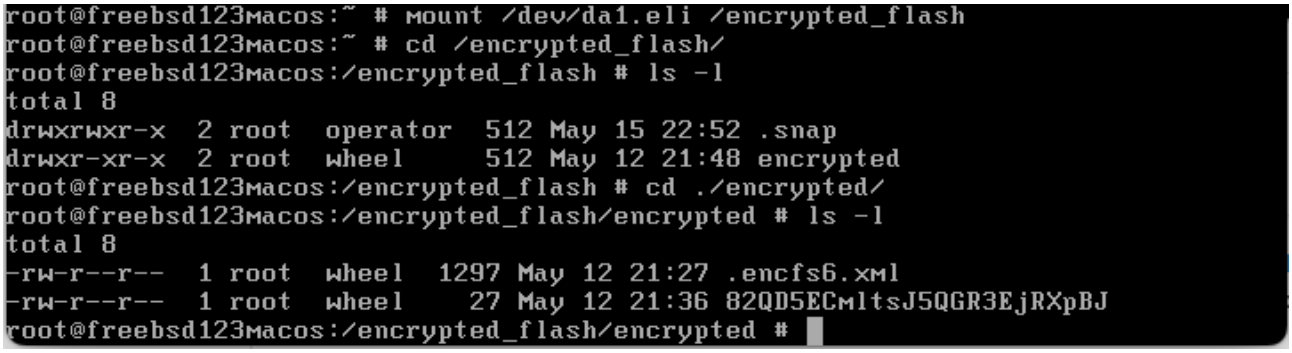

Рисунок 3.10 – Монтування вмісту USB флеш в теку /encrypted_flash в віртуальній машини FreeBSD

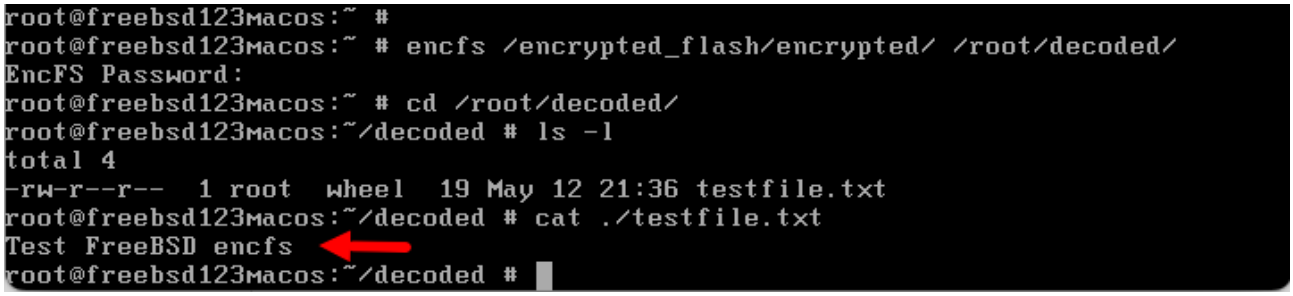

Рисунок 3.11 – Розшифрування та монтування вмісту зашифрованої теки /encrypted в віртуальній машини FreeBSD

Як можна побачити з рисунку 3.11 вміст теки /encrypted, який був зашифрований за допомогою EncFS розшифровано. Вміст файлу файл testfile.txt коректно відображається.

За допомогою SFTP можна відкрити файл testfile.txt в MacOS та працювати з файлом як з локального диску (Рис.3.12). SFTP - це мережевий протокол, який дозволяє безпечно передавати та керувати файлами за допомогою захищеного з'єднання SSH. SFTP надає автентифікацію та шифрування для забезпечення конфіденційності даних що передаються.

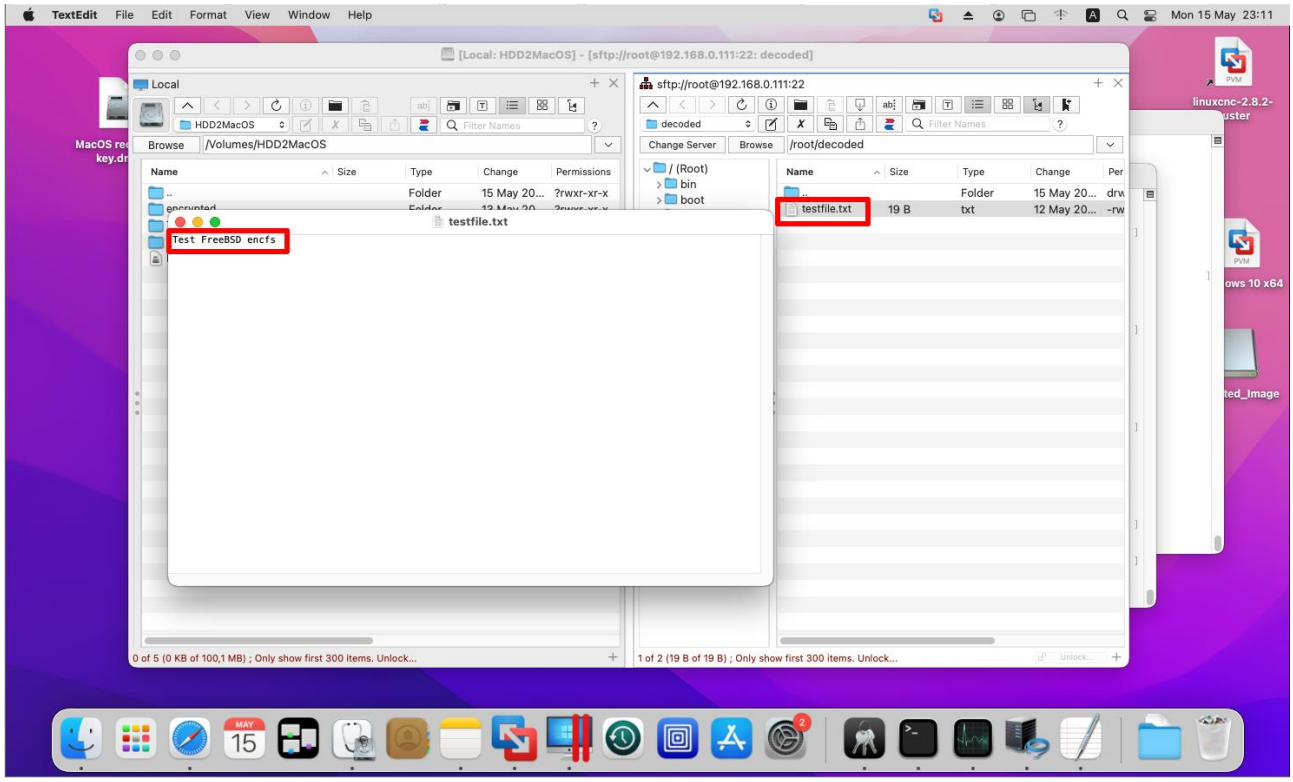

Рисунок 3.12 – Робота з файлом testfile.txt в MacOS за допомогою SFTP

Використання гіпервізора VMware Fusion для створення віртуальної машини з FreeBSD на macOS дозволяє працювати з GELI та EncFS безпосередньо на macOS. Це надає можливість розшифрувати вміст USB флеш і працювати з ним у безпечному середовищі віртуальної машини. Використання GELI та EncFS забезпечує шифрування та захист ваших даних під час зберігання. Крім того, цей підхід дозволяє забезпечити сумісність між macOS та FreeBSD і зручність користування.

Використання віртуалізації є ефективним рішенням для забезпечення безпеки та зручності роботи з шифрованими даними на FreeBSD у середовищі macOS.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Домедична допомога при шоку

Домедична допомога постраждалим при підозрі на шок є важливою процедурою, яку можуть виконувати особи без медичної освіти. Шок - це невідкладний стан, що виникає внаслідок порушення оксигенації тканин організму та може призвести до дисфункції важливих органів та систем.

Ознаки шоку включають бліду, холодну та вологу шкіру, загальну слабкість, неспокій, роздратованість, сухість в роті, спрагу, змінену частоту дихання та свідомість.

Причинами шоку можуть бути масивна зовнішня або внутрішня кровотеча, травми, анафілаксія або серцевий напад.

Наказ Міністерства охорони здоров'я України від 09.03.2022 р. № 441 " Про затвердження порядків надання домедичної допомоги особам при невідкладних станах" встановлює порядки надання домедичної допомоги постраждалим при підозрі на шок [14].

Надання домедичної допомоги постраждалим при підозрі на шок включає такі дії:

1) Переконатися у відсутності небезпеки та, якщо небезпеки немає, переходити до наступного кроку.
2) Заспокоїти постраждалого та пояснити свої дії.
3) Викликати швидку медичну допомогу та слідувати інструкціям диспетчера.
4) Виявити та усунути причину шоку, якщо це можливо.
5) Надати постраждалому протишокове положення:

- перевести постраждалого в горизонтальне положення, якщо це не погіршує його дихання;
- покласти під ноги валик з одягу тощо таким чином, щоб ступні ніг знаходились на рівні його підборіддя;
- підкласти під голову постраждалого одяг/подушку, якщо це не погіршує його дихання;

- вкрити постраждалого ковдрою або покривалом.

6) Забезпечити постійний нагляд за постраждалим до прибуття швидкої медичної допомоги.
7) У разі погіршення стану постраждалого повторно викликати швидку медичну допомогу.
8) Зібрати максимально можливу інформацію про обставини травми та її механізм і передати цю інформацію працівникам швидкої медичної допомоги або диспетчеру.

Якщо постраждалий втратив свідомість до прибуття швидкої медичної допомоги, слід перейти до процедури надання домедичної допомоги дорослим або дітям при раптовій зупинці кровообігу, відповідно до встановленого Порядку.

Виконання цих кроків допоможе забезпечити постраждалому першу необхідну допомогу та зберегти його життя до прибуття медичних фахівців.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

В сучасному інформаційному суспільстві комп'ютери використовуються на робочих місцях велика кількості людей. Ефективна робота з ПК залежить від відповідності організації робочого місця принципам ергономіки. Ергономічна організація робочого місця оператора ПК сприяє комфортній, безпечній та продуктивній роботі.

Один з головних аспектів ергономічної організації робочого місця оператора ПК - це правильне розташування обладнання. Комп'ютерний монітор повинен бути розташований на відстані 50-70 см від очей оператора, з вертикальним кутом огляду близько 15-20 градусів. Клавіатура та миша повинні бути розташовані таким чином, щоб зап'ястя знаходилися в прямому положенні, не перенапружуючись під час роботи. Також важливо, щоб стіл та стілець були підібрані з урахуванням зручності та підтримки правильної позиції тіла оператора.

Стілець з правильною підтримкою спини та регульованою висотою допомагає уникнути напруження спини та шийних м'язів.

Висота стола повинна бути такою, щоб плечі оператора були розслаблені, а передпліччя лежали на столі без напруження. Належне розташування робочої поверхні забезпечує комфортну роботу з клавіатурою та мишкою.

Освітлення є ще одним важливим аспектом ергономічної організації робочого місця. Робоче місце повинно мати достатнє освітлення, яке не створює відблиску на екрані монітора. Краще використовувати природне світло, а також штучне освітлення з регульованою яскравістю. Джерела світла повинні розташовуватись збоку або ззаду від оператора, щоб уникнути тіней та відблиску на робочій поверхні.

Правильна позиція тіла оператора є також важливою у ергономічній організації робочого місця. Оператор повинен сидіти зі спиною прямо, з підтримкою нижньої частини спини. Ноги повинні бути розташовані на підлозі з прогнутими колінами під кутом близько 90 градусів. Руки та зап'ястя повинні бути розташовані в нейтральному положенні, без напруження або перекручування.

До інших вимог ергономіки до організації робочого місця оператора ПК належить забезпечення достатнього простору для розташування обладнання та документів, використання антистатичного покриття для підлоги, що зменшує накопичення статичної електрики, та забезпечення належного провітрювання приміщення.

В законодавчій базі документів, які стосуються вимог безпеки до обладнання та технологічних процесів та враховують ергономічні аспекти у дизайні робочих місць. У якості прикладу можна навести такі нормативні документи:

– ДСТУ ISO 9241-1-2003 Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення.

– ДСТУ 7234:2011 Дизайн та ергономіка.

– ДСТУ 7299:2013 Дизайн та ергономіка. Робоче місце оператора Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки Обладнання виробниче.

– ДСТУ 7950:2015 Дизайн і ергономіка. Робоче місце при виконанні робіт стоячи.

– ДСТУ 7951:2015 Дизайн і ергономіка. Крісло оператора. Загальні ергономічні вимоги Загальні ергономічні вимоги.

– ДСТУ 8604:2015 Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги.

– ДСТУ EN 547-3:2018 Безпека машин. Розміри тіла людини. Частина 3. Антропометричні дані.

Ергономічна організація робочого місця оператора ПК має велике значення для забезпечення комфорту, безпеки та ефективності роботи. Відповідне розташування обладнання, належне освітлення, правильна позиція тіла та інші аспекти ергономічної організації робочого місця сприяють запобіганню втоми, м'язовим напруженням та іншим проблемам, пов'язаним з роботою на ПК. Робота на ергономічно організованому робочому місці підвищує продуктивність, знижує ризик розвитку травм та покращує загальне самопочуття оператора ПК.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи були досліджені та практично застосовані методи шифрування, такі як GELI і EncFS в FreeBSD та Encrypted APFS в macOS. Дані методи шифрування надають засоби для захисту конфіденційної інформації і забезпечення безпеки даних.

GELI є одним з найкращих методів шифрування, доступним у FreeBSD. Він дозволяє створювати зашифровані розділи, що забезпечують захист даних від несанкціонованого доступу. GELI має різні опції налаштування, включаючи використання різних шифрувальних алгоритмів та режимів роботи.

EncFS є ще одним методом шифрування, який доступний як в FreeBSD що дозволяє шифрувати теки та файли окремо від всієї файлової системи. Він створює зашифровані теки, де дані автоматично шифруються при записі і розшифровуються при читанні. EncFS також надає можливість використовувати різні шифрувальні алгоритми та параметри шифрування.

Encrypted APFS - це файлова система APFS (Apple File System), яка підтримує шифрування даних. APFS є основною файловою системою в операційних системах macOS High Sierra і новіших версіях. Шифрування даних у файловій системі APFS забезпечує захист конфіденційності і безпеки файлів шляхом застосування сильного шифрування на рівні файлової системи. Це означає, що всі дані будуть зашифровані під час збереження на диску, і вони будуть розшифровані автоматично, коли буде отримано доступ до них через пароль або інші автентифікаційні дані.

В MacOS є можливість встановити гіпервізор другого типу від різних виробників. Це дає можливість встановлювати віртуальні машини з операційними системи Windows, Linux та Unix та дозволяє запускати програми та інструменти, які не підтримуються MacOS. Також віртуалізація дозволяє працювати з файлами зашифрованими GELI та EncFS в FreeBSD безпосередньо на MacOS через віртуальну машину та SFTP.

Методи шифрування – GELI, EncFS і Encrypted APFS - надають ефективні засоби для захисту даних у FreeBSD і macOS від несанкціонованого доступу.

Вибір методу шифрування залежить від потреб, налаштувань і специфічних вимог щодо безпеки даних в конкретній операційній системі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Graduate Course in Applied Cryptography [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://toc.cryptobook.us/book.pdf
2. Symmetric Cryptography vs Asymmetric Cryptography [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.baeldung.com/cs/symmetric-vs-asymmetric-cryptography
3. Горбенко І.Д., Горбенко Ю.І. Прикладна криптологія. Теорія. Практика. Застосування. Монографія Харків, Видавництво Форт, 2012 р.
4. US Secure Hash Algorithms [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://datatracker.ietf.org/doc/rfc6234/
5. Hybrid Public Key Encryption [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://datatracker.ietf.org/doc/rfc9180/
6. Encrypting Disk [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.freebsd.org/en/books/handbook/disks/#disks-encrypting
7. EncFS [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://github.com/vgough/encfs/blob/master/encfs/encfs.pod
8. GELI [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://man.freebsd.org/cgi/man.cgi?query=geli
9. EncFS [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://man.freebsd.org/cgi/man.cgi?query=encfs
10. Encfsctl [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://man.freebsd.org/cgi/man.cgi?query=encfsctl
11. Use FileVault to encrypt your Mac startup disk [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://support.apple.com/en-us/HT204837
12. File system formats available in Disk Utility on Mac [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу:

https://support.apple.com/my-mm/guide/disk-utility/dsku19ed921c/21.0/mac/12.0

13. VMware Fusion Documentation [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.vmware.com/en/VMware-Fusion/index.html
14. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0356-22#n769