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 забезпечення безпеки операційної системи Linux"

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Сидорович В.Р.
підпис (прізвище та ініціали)

Керівник Александер Марек Богуслав
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 Сидоровичу Віталію Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи забезпечення безпеки операційної системи Linux

Керівник роботи Александер Марек Богуслав Антонович, д.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 12.06.2023

3. Вихідні дані до роботи Вимоги до операційної системи Linux

4. Зміст роботи (перелік питань, які потрібно розробити)
Огляд операційної системи Linux та загроз її безпеці.
Розглянути механізми безпеки в операційній системі Linux
Реалізувати на практичному прикладі налаштування SSH сервера
Реалізувати приклад налаштування фаєрвола nftables
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, мета, задачі. Ключові особливості операційної системи Linux. Архітектура операційної системи Linux. Основні загрози безпеці Linux. Механізми безпеки в операційній системі Linux. Управління доступом. Аутентифікація та авторизація. Мережева безпека. Шифрування даних. Шифрування даних за допомого LUKS. Шифрування даних за допомогою EncFS. Захист від вірусів та шкідливих програм. ClamAV. Rootkit Hunter. Моніторинг та аудит подій. Syslog. AIDE. Реалізація безпекових механізмів в Linux: налаштування сервера SSH. Налаштування SSH-сервера. Налаштування SSH аутентифікації за допомогою ключів. Налаштування SSH аутентифікації за допомогою ключів. Налаштування SSH аутентифікації за допомогою ключів. Налаштування фаєрвола nftables. Динамічне обмеження кількості одночасних з'єднань за допомогою nftable . Перевірка працездатності створених правил для nftables. 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 21.01.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	22.01 – 25.01	Виконано
2.	Підбір джерел для опису операційної системи Linux та загроз її безпеці	27.01 – 10.02	Виконано
3.	Опрацювання джерел в галузі дослідження	10.02 – 07.03	Виконано
4.	Аналіз механізмів захисту ОС Linux	10.03 – 27.03	Виконано
5.	Налаштування SSH сервера	30.03-10.04	Виконано
6.	Налаштування фаєрвола nftables	11.04 – 20.04	Виконано
7.	Оформлення розділу «Операційна система Linux та загрози безпеці»	08.03 –15.03	Виконано
8.	Оформлення розділу «Механізми безпеки в операційній системі Linux»	30.03 – 12.04	Виконано
9.	Практична реалізація безпекових механізмів в Linux	25.04-15.05	Виконано
10.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	16.05 – 25.05	Виконано
11.	Оформлення кваліфікаційної роботи	28.05 – 07.06	Виконано
12.	Нормоконтроль	08.06 – 09.06	Виконано
13.	Перевірка на плагіат	10.06 – 11.06	Виконано
14.	Попередній захист кваліфікаційної роботи	14.06 – 15.06	Виконано
15.	Захист кваліфікаційної роботи	21.06.2022	

Студент _____ Сидорович В.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Александер Марек Богуслав
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи забезпечення безпеки операційної системи Linux // Кваліфікаційна робота ОР «Бакалавр» // Сидорович Віталій Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБ-41 // Тернопіль, 2023 // С. 81 , рис. – 25_, табл. – _-_ , кресл. –28 , додат. – _-__.

Ключові слова: LINUX, ORACLE, LUKS, SSH, NFTABLES, BRUTE FORCE, ENCFS, AIDE, DAC, MAC, ACL, PAM, CLAMAV, ROOTKIT HUNTER, SYSLOG.

Дана кваліфікаційна робота присвячена дослідженню методів та механізмів забезпечення безпеки операційної системи Linux. Вона розглядає основні загрози безпеці Linux, включаючи віруси, шкідливі програми, хакерські атаки та витоки інформації, та описує їхній вплив на безпеку системи. Досліджуються існуючі механізми та методи забезпечення безпеки, зокрема управління доступом, аутентифікація та авторизація, мережева безпека, захист даних, захист від шкідливих програм, а також методи моніторингу та аудиту подій. Також проведена практична реалізація різних механізмів безпеки. Результатом кваліфікаційної роботи є практичні рекомендації та приклади для забезпечення безпеки операційної системи Linux в реальних умовах. Практична реалізація різних варіантів покращення безпеки є важливим кроком у забезпеченні безпеки операційної системи Linux. Це дозволяє перевірити ефективність запропонованих методів та механізмів захисту в реальних умовах.

ANNOTATION

Methods of securing the Linux operating system // Thesis of educational level "Bachelor" // Sydorovych Vitalii // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, СБ-41 group // Ternopil, 2023 // P. 81, fig. -25, table. - _-_ , chair. - _28__ , added. -_-__.

Keywords: LINUX, ORACLE, LUKS, SSH, NFTABLES, BRUTE FORCE, ENCFS, AIDE, DAC, MAC, ACL, PAM, CLAMAV, ROOTKIT HUNTER, SYSLOG.

This qualifying work is devoted to the study of methods and mechanisms for ensuring the security of the Linux operating system. It reviews the main threats to Linux security, including viruses, malware, hacks, and information leaks, and describes their impact on system security. Existing security mechanisms and methods are examined, including access control, authentication and authorization, network security, data protection, protection against malware, as well as methods for monitoring and auditing events. The practical implementation of various security mechanisms has also been made. The result of the qualification work are practical recommendations and examples of ensuring the security of the Linux operating system in real conditions. The practical implementation of various options for improving security is an important step in ensuring the security of the Linux operating system. This allows you to test the effectiveness of the proposed methods and protection mechanisms in real conditions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GPL	—	General Public License
VPN	—	Virtual Private Network
IPv4	—	Internet Protocol version 4
IPv6	—	Internet Protocol version 6
TCP/IP	—	Transmission Control Protocol/Internet Protocol
UDP	—	User Datagram Protocol
DNS	—	Domain Name System
DDoS	—	Distributed Denial of Service
DHCP	—	Dynamic Host Configuration Protocol
VLAN	—	Virtual Local Area Network
SSH	—	Secure Shell
TCP	—	Transmission Control Protocol
HTTP	—	Hypertext Transfer Protocol Secure
PAM	—	Pluggable Authentication Modules
ICMP	—	Internet Control Message Protocol
DAC	—	Discretionary Access Control
MAC	—	Mandatory Access Control
ACL	—	Access Control List
UID	—	User Identifier
GID	—	Group Identifier
SELinux	—	Security-Enhanced Linux)
SSL/TLS	—	Secure Sockets Layer/Transport Layer Security
SSLv3	—	Secure Sockets Layer version 3
TLSv1.0	—	Transport Layer Security version 1.0
TLSv1.1	—	Transport Layer Security version 1.1
TLSv1.2	—	Transport Layer Security version 1.2
TLSv1.3	—	Transport Layer Security version 1.3
AES	—	Advanced Encryption Standard

ClamAV — Clam AntiVirus

AIDE — Advanced Intrusion Detection Environment

ВСТУП

Операційна система Linux, заснована на відкритому вихідному коді та розповсюджується безкоштовно, використовується широкою спільнотою користувачів та має значний вплив на сучасний світ інформаційних технологій. Linux використовується в різноманітних пристроях, від серверів до мобільних пристроїв і вбудованих систем, завдяки своїм перевагам у надійності, безпеці та масштабованості.

Забезпечення безпеки операційної системи Linux є надзвичайно важливою задачею в сучасному інформаційному середовищі. Захист від різних загроз, таких як віруси, шкідливі програми, хакерські атаки та витоки інформації, є критично важливим для збереження конфіденційності, цілісності та доступності даних.

Метою цієї кваліфікаційної роботи є дослідження методів та механізмів, які забезпечують безпеку операційної системи Oracle Linux. Операційна система Oracle Linux, розроблена компанією Oracle Corporation, є однією з важливих альтернатив для підприємств, що шукають надійну та безпечну платформу для своїх інформаційних потреб.

У цьому дослідженні ми ставимо перед собою наступні завдання:

1) Розглянути основні загрози безпеці операційної системи Linux.

2) Дослідити основні існуючі механізми та методи забезпечення безпеки в Linux, зокрема управління доступом, аутентифікацію та авторизацію, мережеву безпеку, захист даних, захист від шкідливих програм.

3) Дослідити основні існуючі методи моніторингу та аудити подій.

4) Здійснити практичну реалізацію різних механізмів безпеки.

Також ця кваліфікаційна робота має на меті поглиблене дослідження та аналіз проблем безпеки операційної системи Linux, а також розробку практичних рекомендацій та прикладів для забезпечення безпеки в реальних умовах.

Однак, важливо зазначити, що безпека є постійним процесом, і не існує жодного абсолютно захищеного середовища. Тому розуміння принципів безпеки

та використання ефективних методів та механізмів захисту є ключовими аспектами для забезпечення безпеки операційної системи Linux.

1 ОПЕРАЦІНА СИСТЕМА LINUX ТА ЗАГРОЗИ БЕЗПЕЦІ

1.1 Огляд операційної системи Linux

Linux - це вільна та відкрита операційна система, яка використовується в різних пристроях, від персональних комп'ютерів до серверів, мобільних пристроїв та вбудованих систем. Операційна система Linux ґрунтується на ядрі Linux, розробленому Лінусом Торвальдсом у 1991 році. Вона отримала широку популярність завдяки своїй стабільності, безпеці та гнучкості [1].

Деякі ключові особливості операційної системи Linux:

Вільна та відкрита.

Linux поширюється під ліцензією GNU GPL, що дає користувачам право використовувати, змінювати та поширювати її безкоштовно. Кожен користувач має можливість використовувати операційну систему без будь-яких обмежень. Він може встановлювати Linux на будь-яку кількість комп'ютерів, а також використовувати його для будь-яких цілей - від особистих комп'ютерів до серверів та вбудованих систем.

Окрім того, ліцензія GPL надає користувачам право змінювати та вдосконалювати вихідний код Linux. Це означає, що будь-який користувач може адаптувати систему до своїх потреб, виправити помилки або додати нові функціональні можливості. Ці зміни можна поширювати та спільно використовувати з іншими користувачами, що сприяє розвитку та вдосконаленню операційної системи.

Одним із ключових аспектів відкритості Linux є спільнота розробників та користувачів, яка активно працює над вдосконаленням системи. Ця спільнота сприяє обміну знаннями, вирішенню проблем, розробці нових функцій та забезпеченню безпеки системи.

Різноманітність дистрибутивів.

Існує велика кількість дистрибутивів Linux, таких як Ubuntu, Fedora, Debian, CentOS, Oracle Linux та багато інших. Кожен з них має свої особливості,

спрямовані на різні типи користувачів та сценарії використання. Кожен дистрибутив має свою власну філософію, цільову аудиторію та спрямованість.

Один з найпопулярніших дистрибутивів - Ubuntu, спрямований на використання в домашньому, офісному та серверному середовищі. Він має простий та зручний інтерфейс, широкий спектр програмного забезпечення та підтримку спільноти.

Fedora є дистрибутивом, який акцентується на новітніх технологіях та інноваціях. Він часто використовується в сфері розробки програмного забезпечення та має швидкий та стабільний релізний цикл.

Debian відомий своєю стабільністю та безпекою. Він використовується як основа для багатьох інших дистрибутивів та застосовується як серверна платформа.

Oracle Linux, базується на відкритому коді Red Hat Enterprise Linux (RHEL) і призначений для використання в корпоративному середовищі. Він має довготривалу підтримку. Надійний та безпечний в роботі.

Крім цих дистрибутивів, існує безліч інших варіантів, які орієнтовані на різні сфери використання, такі як наукові дослідження, безпека, мобільні пристрої та інші. Користувачі можуть вибрати дистрибутив, що найкраще відповідає їхнім потребам, забезпечуючи оптимальну функціональність та зручність використання.

Багатозадачність.

Linux дозволяє виконувати багато завдань одночасно. Операційна система здатна керувати багатопотоковими процесами та розподіляти ефективно ресурси. Linux використовує планувальник задач для керування виконанням процесів. Планувальник визначає пріоритети процесів та розподіляє їх на доступних процесорних ядрах. Це дозволяє оптимально використовувати обчислювальні ресурси системи та забезпечує плавну та швидку роботу багатозадачних програм.

Багатозадачність в Linux також включає можливість запуску та управління різними процесами в фоновому режимі. Наприклад, системні служби, мережеві сервіси, процеси резервного копіювання та інші завдання можуть працювати

незалежно від користувача, що дозволяє виконувати різні завдання одночасно без шкоди для основних робочих процесів.

Крім того, Linux підтримує розподілену роботу та може працювати у середовищі з декількома вузлами або комп'ютерами, дозволяючи розподіляти завдання та обчислювальні ресурси між ними. Це особливо важливо в сучасних серверних та кластерних середовищах, де Linux забезпечує можливість ефективного використання ресурсів та масштабування обчислювальних потужностей.

Багатозадачність і розподілена робота в Linux сприяють забезпеченню продуктивності, ефективності та масштабованості системи, що робить його привабливим вибором для різних типів завдань та сценаріїв використання.

Підтримка мережі.

Linux має потужні можливості мережевої взаємодії. Він може функціонувати як сервер або клієнт у різних мережевих середовищах, таких як Інтернет.

В якості сервера, Linux може надавати різноманітні мережеві служби, такі як веб-сервери (наприклад, Apache, Nginx), поштові сервери (наприклад, Postfix, Sendmail), файлообмінні сервери (наприклад, Samba, FTP-сервери), бази даних (наприклад, MySQL, PostgreSQL) та багато інших. Це дозволяє використовувати Linux для розгортання різноманітних мережевих сервісів та додатків.

Як клієнт, Linux надає засоби для з'єднання та взаємодії з різними типами мереж, включаючи Інтернет. В Linux вбудовані клієнти для протоколів зв'язку, таких як HTTP/HTTPS (наприклад, браузери Mozilla Firefox), SSH (наприклад, OpenSSH) та багато інших. Це дозволяє користувачам Linux використовувати різноманітні мережеві сервіси та додатки, що доступні в Інтернеті та локальних мережах.

Linux також має широку підтримку мережевих протоколів та технологій, включаючи IPv4 та IPv6, TCP/IP, UDP, DNS, DHCP, VPN, VLAN, Wi-Fi та багато інших. Це дозволяє ефективно використовувати Linux у різних мережевих середовищах та забезпечує його сумісність з різними мережевими пристроями та інфраструктурою.

Завдяки своїм потужним можливостям мережевої взаємодії, Linux став популярним вибором для серверів та мережевих пристроїв. Відкритий характер операційної системи дозволяє розробникам та адміністраторам налаштовувати та розширювати мережеві функції Linux згідно зі своїми потребами.

Безпека.

Операційна система Linux славиться своєю високою безпекою. Завдяки великій спільноті розробників, які активно працюють над виявленням та виправленням вразливостей, Linux забезпечує високий рівень захисту від зловмисних атак.

Існує кілька факторів, що призводять до високого рівня безпеки в Linux:

- Відкритий код. Один з головних аспектів безпеки Linux полягає у його відкритості. Оскільки вихідний код Linux доступний для всіх, спільнота розробників постійно перевіряє його на вразливості та помилки. Багато очей, що ретельно переглядають код, сприяє виявленню та виправленню проблем швидше, ніж у випадку закритих комерційних операційних систем;
- Швидкі виправлення. Завдяки активній спільноті розробників і користувачів, вразливості, які виявляються в Linux, швидко виправляються шляхом надання патчів та оновлень безпеки. Це дозволяє оперативно реагувати на нові загрози та забезпечує постійну підтримку безпеки;
- Управління привілеями. Linux має досконалу систему керування привілеями, яка дозволяє точно визначити права доступу до різних системних ресурсів. Користувачі та програми можуть мати обмежений доступ лише до необхідних ресурсів, що допомагає запобігати несанкціонованій діяльності;
- Вбудовані захисти. Linux має вбудовані механізми безпеки, такі як обмеження привілеїв, мережеві брандмауери, шифрування файлів та комунікацій, аутентифікацію користувачів, контроль доступу та аудит безпеки. Ці механізми допомагають уникнути багатьох типів атак та забезпечують захист інформації;

- Віддалений моніторинг та аудит. Linux надає засоби для віддаленого моніторингу та аудиту безпеки, що дозволяє адміністраторам системи стежити за подіями, виявляти підозрілу активність та реагувати на потенційні загрози.

Разом ці фактори сприяють створенню високого рівня безпеки в операційній системі Linux. Однак, важливо пам'ятати, що безпека залежить не тільки від самої операційної системи, але й від належного конфігурування та знання користувачем безпечних практик.

Консольні і графічні інтерфейси.

Linux надає користувачам можливість вибору між роботою у текстовому режимі (консоль) або графічному режимі.

Кожен з цих режимів має свої переваги та призначення, а також надає різні графічні оболонки для сприяння комфортному користувацькому досвіду:

- Консольний режим. У консольному режимі користувач працює у текстовому середовищі, використовуючи командний рядок. Цей режим забезпечує повний контроль та гнучкість у виконанні завдань, дозволяє автоматизувати рутинні процеси за допомогою скриптів та команд, і має низькі вимоги до системних ресурсів. Консольний режим особливо корисний для системних адміністраторів, розробників та досвідчених користувачів, які шукають максимальну контрольованість і ефективність;
- Графічний режим. Графічний режим надає зручне інтерактивне середовище, що використовує візуальні елементи, такі як вікна, панелі, меню та іконки. В Linux існує багато графічних оболонок (графічних інтерфейсів), які забезпечують різні стилі та функціональні можливості. Найпоширенішими графічними оболонками є GNOME, KDE, Xfce тощо. Графічний режим зручний для користувачів, які більше звикли до візуального інтерфейсу та бажають простоти у використанні.

Завдяки можливості вибору між консольним та графічним режимами, користувачі Linux можуть налаштувати робоче середовище, що найкраще відповідає їхнім потребам та вподобанням. Крім того, можна комбінувати обидва

режими для досягнення оптимального балансу між продуктивністю та зручністю використання.

Розширені можливості.

Linux має широкий спектр програм та інструментів для розробки, системного адміністрування, баз даних, мультимедіа, графіки тощо. Користувачі можуть використовувати його для різних завдань, від веб-серверів до наукових досліджень.

1.2 Архітектура операційної системи Linux

Архітектура Linux включає різні компоненти та підсистеми, які співпрацюють для забезпечення роботи системи [2]. Нижче наведено опис кожного з компонентів архітектури Linux:

Ядро (Kernel).

Ядро Linux є центральною частиною операційної системи. Воно забезпечує основні функції, такі як керування пам'яттю, планування процесів, керування файловою системою, мережеві операції та інтерфейси для взаємодії з апаратним забезпеченням.

Драйвери пристроїв (Device Drivers).

Драйвери пристроїв взаємодіють з апаратним забезпеченням і дозволяють ядру спілкуватися з різними пристроями, такими як клавіатура, миша, диски, мережеві карти та інші. Вони забезпечують стабільну роботу пристроїв і передачу даних між пристроями та ядром. Недостатня безпеки в модулях ядра та драйверах може призвести до вразливостей, які можуть бути використані зловмисниками для злому системи або отримання несанкціонованого доступу.

Системний виклик (System Call).

Системні виклики - це механізми, які дозволяють програмам взаємодіяти з ядром. Вони надають доступ до функцій операційної системи, таких як керування файлами, мережеві операції, керування процесами тощо. Системні виклики забезпечують безпеку, контроль доступу та стабільність операційної системи. Зловмисники можуть спробувати використати системні виклики для

злому системи, отримання несанкціонованого доступу або виконання шкідливих операцій. Тому важливо забезпечити контроль та перевірку системних викликів для запобігання можливим атакам.

Процеси та планувальник (Processes and Scheduler).

Операційна система Linux підтримує багатозадачність, що дозволяє багатьом процесам виконуватися одночасно. Планувальник вирішує, який процес отримає доступ до процесора та в якому порядку. Він забезпечує ефективне розподілення ресурсів системи та керування виконанням процесів.

Файлова система (File System).

Файлова система Linux організовує збереження та доступ до файлів та директорій. Вона дозволяє користувачам створювати, зчитувати, записувати та видаляти файли. Linux підтримує різні типи файлових систем, такі як ext4, XFS та інші. Вибір правильної файлової системи може впливати на безпеку системи, оскільки різні файлові системи мають свої особливості щодо стійкості до відновлення даних, контролю доступу та криптографічного захисту.

Мережевий стек (Networking Stack).

Linux має потужний мережевий стек, який забезпечує мережеву комунікацію. Він підтримує протоколи мережі, такі як TCP/IP, UDP, ICMP та інші. Мережевий стек дозволяє передавати дані через мережу. Вразливості в мережевому стеку можуть призвести до атак на мережеву безпеку, злому системи або перехопленню та зміні даних при передачі.

1.3 Основні загрози безпеці Linux

Linux, як і будь-яка інша операційна система, піддається впливу різноманітних загроз, які можуть викликати порушення безпеки, доступ до конфіденційної інформації та завдати шкоди системі [3].

Нижче наведено опис деяких основних загроз безпеці Linux.

Вразливості програмного забезпечення.

Вразливості програмного забезпечення можуть бути використані зловмисниками для злому системи. Це можуть бути вразливості в ядрі Linux,

драйверах пристроїв, програмах або сервісах, які виконуються на системі. Зловмисники можуть експлуатувати ці вразливості, щоб отримати несанкціонований доступ або запустити шкідливий код на системі.

Злам паролів.

Зловмисники можуть спробувати зламати паролі користувачів для отримання несанкціонованого доступу до системи. Вони можуть використовувати методи, такі як перебір паролів.

Мережеві атаки.

Linux-сервери, які підключені до мережі Інтернет, можуть бути піддані різноманітним мережевим атакам. Це можуть бути атаки DDoS, зміна трафіку, використання вразливостей мережевих протоколів, перехоплення пакетів. Мережеві атаки можуть призвести до блокування сервісу, розкриття конфіденційної інформації або викрадення даних.

Соціальний інжиніринг.

Соціальний інжиніринг є методом, в якому зловмисники використовують маніпуляції та обман, щоб отримати доступ до системи. Вони можуть використовувати фішингові атаки, представляти себе за легітимних користувачів або отримати конфіденційну інформацію від користувачів. Соціальний інжиніринг є вихідним пунктом для подальших атак на систему.

Недостатні оновлення та патчі.

Недостатні оновлення операційної системи Linux можуть призвести до невчасного виправлення вразливостей. Зловмисники можуть експлуатувати ці вразливості, які вже відомі, для злому системи. Регулярне оновлення і застосування патчів є важливим для забезпечення безпеки Linux.

Віруси та шкідливі програми.

Віруси та шкідливі програми є одними з поширених загроз безпеці для операційної системи Linux. Вони можуть виконувати шкідливі дії, такі як руйнування даних, перехоплення конфіденційної інформації або створення "задніх дверей" для зловмисників.

2 МЕХАНІЗМИ БЕЗПЕКИ В ОПЕРАЦІЙНІЙ СИСТЕМІ LINUX

2.1 Управління доступом

У операційній системі Linux існує ряд механізмів безпеки, які забезпечують управління доступом до ресурсів системи. У цьому розділі ми розглянемо основні методи та механізми управління доступом в Linux, зокрема систему прав доступу до файлів (DAC), обов'язковий контроль доступу (MAC) і систему розширених прав доступу до файлів та каталогів (ACL) [4].

2.1.1 Система прав доступу до файлів (DAC)

DAC або система дискреційного контролю доступу - це механізм управління доступом в операційній системі Linux, який базується на правах, які призначені власником ресурсу. DAC визначає, які користувачі або групи користувачів мають право доступу до файлів, директорій та інших ресурсів в системі [4].

Коли користувач створює файл або директорію, встановлюються права доступу, які визначають, що можна робити з цими об'єктами. Права доступу включають права для власника (u), групи користувачів, до якої власник належить (g), і всіх інших користувачів (o). За допомогою команди chmod можна змінювати ці права.

Типові права доступу включають:

- Читання (r) - дозволяє читати (переглядати) вміст файлу або директорії.
- Запис (w) - дозволяє змінювати (редагувати) вміст файлу або директорії, а також створювати або видаляти файли в директорії.
- Виконання (x) - дозволяє виконувати (запускати) файл або переходити до директорії.

На рисунку 2.1 показано вивід команди `ls -l` де можна побачити встановлені права доступу до тек та файлів для власника, групи та інших користувачів.

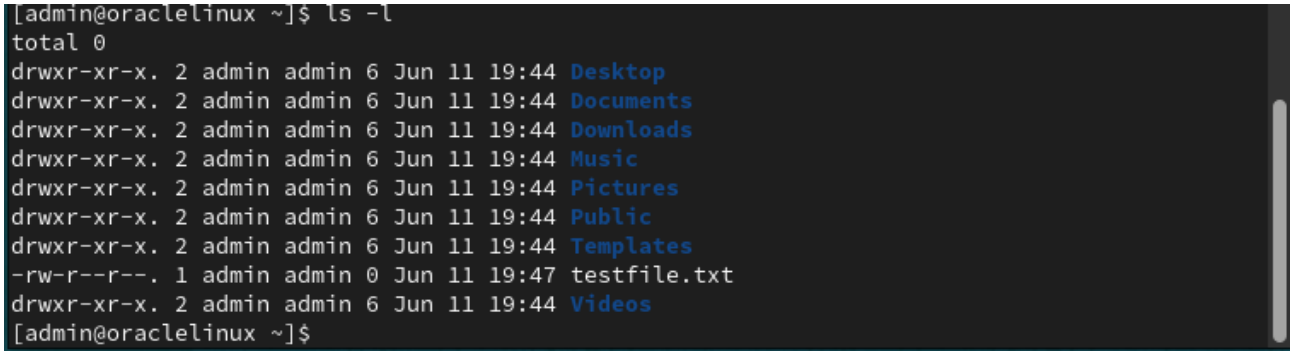

Рисунок 2.1 – Вивід команди `ls -l`

За допомогою прав доступу DAC можна обмежувати доступ до ресурсів і встановлювати, які дії можуть бути виконані з цими ресурсами. Однак DAC має свої обмеження, так як власник ресурсу може встановлювати права доступу і змінювати їх, що може призвести до потенційних проблем безпеки.

UID та GID - це числові ідентифікатори, які використовуються в операційній системі Linux для ідентифікації користувачів та груп.

UID - це унікальний ідентифікатор, призначений кожному користувачеві в системі. Він використовується для ідентифікації користувача під час аутентифікації та контролю доступу до ресурсів. Кожен користувач в системі має свій власний UID, який дозволяє системі відрізняти його від інших користувачів. Зазвичай, UID 0 належить спеціальному користувачеві root, який має повні привілеї в системі.

GID - це ідентифікатор групи, який призначений кожній групі користувачів в системі. Він використовується для організації користувачів в групи та контролю доступу до ресурсів на основі групових прав. Кожен користувач може належати до однієї або кількох груп, і кожна група має свій унікальний GID. Групи дозволяють легко управляти правами доступу до файлів та директорій для кількох користувачів, які мають спільні потреби.

Коли користувач створює файл або директорію, встановлюються не тільки права доступу (через DAC), але також визначається UID власника та GID групи власника цього ресурсу. Ці значення використовуються для визначення прав доступу до цього ресурсу. Інші користувачі та групи мають свої власні UID та GID, і їхні права доступу встановлюються відповідно.

UID та GID важливі для безпеки в Linux, оскільки вони визначають, які користувачі та групи мають доступ до конкретних ресурсів і які дії вони можуть виконувати з цими ресурсами. Вірне призначення і управління UID та GID є важливою складовою управління доступом та забезпечення безпеки в операційній системі Linux.

Команда chown використовується в операційній системі Linux для зміни власника файлу або директорії. За допомогою цієї команди можна змінювати UID та GID власника об'єкта.

Для додаткового контролю доступу і забезпечення вищого рівня безпеки, в Linux також використовуються системи обов'язків (MAC) та системи розширених прав доступу до файлів та каталогів (ACL), про які буде згадано в наступних розділах.

2.1.2 Обов'язковий контроль доступу (MAC)

MAC - це система обов'язкового контролю доступу, яка використовується в операційній системі Linux для забезпечення додаткового рівня безпеки [4]. У порівнянні зі стандартною системою прав доступу до файлів (DAC), де власник файлу вирішує, хто має доступ до файлу і які дії можуть бути виконані, система MAC базується на політиках безпеки, встановлених адміністратором системи.

Основною ідеєю MAC є призначення міток безпеки для об'єктів, таких як файли, процеси, сокети тощо, а також для суб'єктів, таких як користувачі та групи. Ці мітки визначають рівень довіри та доступу до об'єктів. Кожна операція, яка виконується в системі, перевіряється на відповідність правилам політики безпеки, і доступ до об'єктів може бути дозволений або заборонений на основі цих правил.

У Linux одним з найбільш відомих і популярних реалізацій системи MAC є SELinux. SELinux використовує мітки безпеки для файлів, процесів та ресурсів системи, і має розширені можливості для контролю доступу до системних ресурсів.

SELinux реалізує систему обов'язкового контролю доступу, де кожен об'єкт і суб'єкт в системі має свою мітку безпеки. Ці мітки визначають рівень доступу до ресурсів і контролюють, які операції можуть бути виконані над цими ресурсами. SELinux додатково до системи DAC дозволяє досить гнучко налаштовувати правила доступу для окремих об'єктів і суб'єктів системи.

Використання SELinux дозволяє підвищити безпеку системи шляхом обмеження можливостей зловмисників або недбайливих користувачів. SELinux забезпечує контроль доступу на рівні ядра операційної системи, що дозволяє блокувати шкідливі або небезпечні операції ще до їх виконання.

Однак використання SELinux вимагає налагодження політик безпеки та ретельного аналізу прав доступу до ресурсів. Неправильне налаштування може призвести до проблем з доступом до файлів, процесів або мережевих ресурсів, тому важливо мати розуміння принципів роботи SELinux та проводити налагодження з обережністю.

На рисунку 2.2 показано вивід команди `id` на якому можна інформацію про користувача `admin`.

```
[admin@oraclelinux ~]$ id
uid=1000(admin) gid=1000(admin) groups=1000(admin),10(wheel) context=unconfined_u:unconfined
_r:unconfined_t:s0-s0:c0.c1023
[admin@oraclelinux ~]$
```

Рисунок 2.2 – Вивід команди `id`

У вказаному рядку представлена інформація про користувача з UID 1000, ім'ям "admin" та GID 1000, також з назвою "admin". Користувач також належить до групи з GID 10, що називається "wheel". В даному контексті, контекст безпеки (security context) вказує на "unconfined_u" (неконтрольований користувач), "unconfined_r" (неконтрольована роль), "unconfined_t" (неконтрольований тип) з діапазоном "s0-s0:c0.c1023".

Група з GID 10, що називається "wheel", використовується для адміністративних цілей. Користувачі, які належать до групи "wheel", мають підвищені привілеї та доступ до деяких системних операцій.

Контекст безпеки (security context) вказує на обмеження та правила, які застосовуються до процесу або ресурсу в системі SELinux. В даному випадку, контекст безпеки "unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023"

означає, що користувач та процес, пов'язані з цим UID, GID та контекстом, не обмежені правилами SELinux і мають відносно вільний доступ до ресурсів.

Для демонстрації принципу роботи SELinux створимо нового користувача `limituser` з контекстом безпеки guest_u (Рис.2.3).

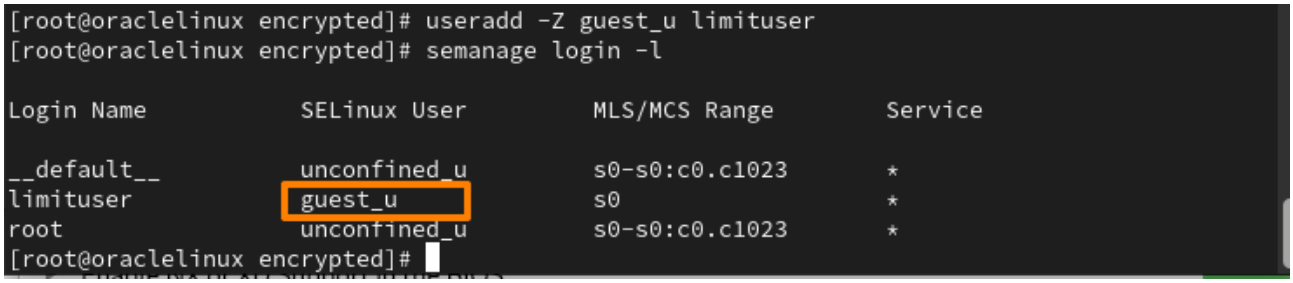

Рисунок 2.3 – Створення нового користувача limituser з контекстом безпеки guest_u

В цьому прикладі, команда `semanage login -l` використовується для переліку SELinux-контекстів, пов'язаних зі списком користувачів системи. Кожен рядок представляє окремого користувача та його SELinux-контекст.

`Login Name` - це ім'я користувача системи.

`SELinux User` - це значення вказує на SELinux-контекст, який призначений для цього користувача.

`MLS/MCS Range` - це діапазон значень, який вказує на рівень чутливості або мультирівневий контроль доступу. У цьому прикладі, значення s0 вказує на стандартний рівень чутливості без мультирівневого контролю доступу.

`Service` - це сервіс або роль, яку можна пов'язати з цим користувачем. Зірочка (*) позначає, що цей користувач не пов'язаний з жодним конкретним сервісом або роллю.

SELinux контекст `guest_u` використовується для обмеження привілеїв і доступів користувача до системних ресурсів. В контексті `guest_u` встановлені обмеження, які надають користувачу лише обмежений доступ до системи і дозволяють йому виконувати лише деякі дії.

За допомогою контексту `guest_u`, SELinux може контролювати доступ користувача до файлів, каталогів, процесів та інших системних ресурсів. Контекст `guest_u` може бути використаний для створення ізольованих середовищ, де користувачам надається доступ лише до обмеженого набору

функцій або даних, щоб забезпечити безпеку і відокремлення від інших користувачів і ресурсів системи.

SELinux контекст `guest_u` включає в себе не тільки обмеження на доступ до системних ресурсів, але також обмежує доступ до мережі для користувачів з цим контекстом (Рис.2.4).

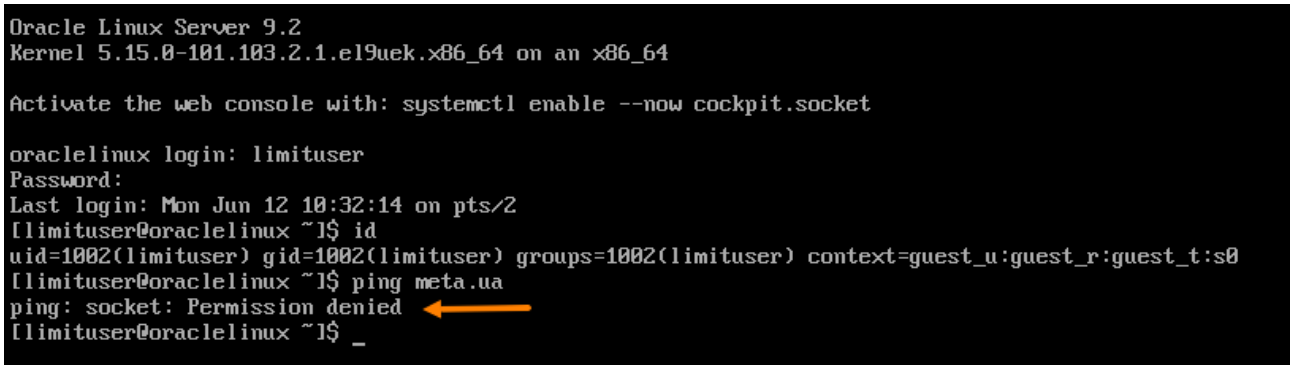

Рисунок 2.4 – Перевірка обмежень доступу до мережі для користувача limituser

Використання системи MAC дозволяє забезпечити більш жорсткий контроль доступу до ресурсів системи, запобігти поширенню зловмисних програм і зменшити ризик зловживання привілеями. Однак вона також вимагає налагодження політики безпеки та ретельного аналізу прав доступу до ресурсів, що може бути складним завданням для адміністраторів системи.

2.1.3 Система розширених прав доступу до файлів та каталогів (ACL)

Система розширених прав доступу до файлів та каталогів (ACL) в Linux дозволяє більш гнучко налаштовувати права доступу до файлів та каталогів, включаючи додаткові рівні контролю, які виходять за межі стандартної системи прав доступу (DAC).

У стандартній системі прав доступу в Linux (DAC) використовуються три основні права доступу: власник (owner), група (group) та інші користувачі (others). Кожному користувачеві або групі присвоюються певні права на читання (read), запис (write) та виконання (execute) файлу або каталогу.

ACL додає додаткові рівні контролю доступу, дозволяючи встановлювати специфічні права доступу для конкретних користувачів або груп, незалежно від стандартної системи прав доступу [4]. Це дає більш гнучкий та деталізований контроль над доступом до файлів та каталогів.

Для використання ACL необхідно мати файлову систему, що підтримує цю функцію, а також ядро Linux, компільоване з підтримкою ACL. Більшість сучасних дистрибутивів Linux підтримують ACL за замовчуванням.

За допомогою ACL можна встановлювати індивідуальні права доступу для кожного користувача та групи, а також надавати спеціальні привілеї, наприклад, дозвіл на читання, запис або виконання, встановлення специфічних масок доступу та інших параметрів.

Команди, що використовуються для роботи з ACL, включають `getfacl` (отримання списку ACL для файлу або каталогу), `setfacl` (встановлення ACL для файлу або каталогу) та `chacl` (зміна ACL).

На рисунку 2.5 показано встановлення та перевірки ACL на файлі testfile.txt

```
[admin@oraclelinux ~]$ setfacl -m u:vitaliy:rwx testfile.txt
[admin@oraclelinux ~]$ getfacl testfile.txt
# file: testfile.txt
# owner: admin
# group: admin
user::rw-
user:vitaliy:rwx
group::r--
mask::rwx
other::r--

[admin@oraclelinux ~]$
```

Рисунок 2.5 – Встановлення та перевірки ACL на файлі `testfile.txt`

У цьому прикладі ми надаємо користувачу з ім'ям `vitaliy` права на читання (r), запис (w) та виконання (x) файлу `testfile.txt`.

Також можна здійснити видалення ACL з файлу:

```
#setfacl -b testfile.txt
```

Ця команда видаляє всі ACL з файлу testfile.txt, повертаючи його до стандартної системи прав доступу (DAC).

Використання системи розширених прав доступу до файлів та каталогів дозволяє докладно налаштовувати контроль доступу до ресурсів в Linux і забезпечує більшу гнучкість та безпеку управління файловою системою.

2.2 Аутентифікація та авторизація

Аутентифікація та авторизація є важливими аспектами безпеки в операційній системі Linux.

Аутентифікація - це процес перевірки і підтвердження ідентичності користувача або системи. Це важливий етап в безпеці інформаційних систем, який дозволяє визначити, чи має особа або система право отримати доступ до певних ресурсів або виконувати певні дії.

Аутентифікація зазвичай базується на представленні ідентифікаційних даних, таких як ім'я користувача і пароль, сертифікат, біометричні дані (відбитки пальців, розпізнавання обличчя тощо) або інші фактори ідентифікації. Під час аутентифікації система перевіряє ці дані на відповідність збереженим або налаштованим даним.

Аутентифікація в Linux може використовувати різні методи і протоколи, включаючи парольну аутентифікацію, ключі SSH, сертифікати X.509 та багато інших. Кожен метод має свої особливості і рівень безпеки, і вибір конкретного методу залежить від потреб і налаштувань системи або додатків.

Авторизація - це процес визначення прав доступу і контролю за доступом після успішної аутентифікації користувача або системи. Після того, як ідентичність користувача підтверджена, авторизація визначає, які дії, операції, ресурси або функції можуть бути доступні для цього користувача в рамках системи або додатка.

Авторизація зазвичай базується на ролях, групах або правилах, встановлених в системі. Користувачеві присвоюються певні права, які визначають, до яких ресурсів він має доступ і які операції може виконувати. Наприклад, адміністратор системи може мати повний доступ до всіх ресурсів і

функцій, тоді як звичайний користувач може мати обмежений доступ та можливості.

У Linux для авторизації використовується система контролю доступу, яка включає механізми, такі як PAM (Pluggable Authentication Modules).

2.2.1 Парольна аутентифікація.

Парольна аутентифікація - це один з найпоширеніших методів аутентифікації, який використовується для перевірки ідентичності користувача на основі пароля [4]. Користувачі обирають пароль, який вони повинні ввести для підтвердження своєї ідентичності під час входу в систему або доступу до обмежених ресурсів.

Процес парольної аутентифікації зазвичай включає наступні етапи:

1) Введення пароля. Користувач вводить свій пароль під час процедури входу в систему або при доступі до обмежених ресурсів.

2) Хешування пароля. Пароль перетворюється на хешоване значення, яке представляє собою унікальний код, отриманий з пароля за допомогою хеш-функції. Хеш-функція перетворює пароль в нерозбірливий хеш, що не може бути зворотно перетворений на початковий пароль.

3) Порівняння хешів. Збережений хеш пароля порівнюється з хешем, отриманим під час введення пароля користувачем. Якщо хеші співпадають, то пароль вважається правильним і процес аутентифікації успішним.

Важливо враховувати наступні аспекти парольної аутентифікації для забезпечення безпеки:

1) Складність пароля. Рекомендується використовувати складні паролі, які складаються з комбінації великих і малих літер, цифр і спеціальних символів. Це допомагає ускладнити процес підбору пароля зловмисниками.

2) Політики паролів. Система може встановлювати правила та обмеження щодо довжини пароля, використання певних символів або періодичної зміни пароля. Це сприяє збільшенню безпеки парольної аутентифікації.

3) Захист хешів. Важливо захищати збережені хеші паролів від несанкціонованого доступу. Зазвичай використовується метод солі (salt), коли до пароля додається додатковий рандомний рядок перед хешуванням. Це ускладнює підбір пароля зловмисниками шляхом використання попередньо обчислених хешів.

Паролі зберігаються в зашифрованому вигляді у файлі `/etc/shadow`.

На рисунку 2.6 показано хеш пароля користувача `vitaliy`.

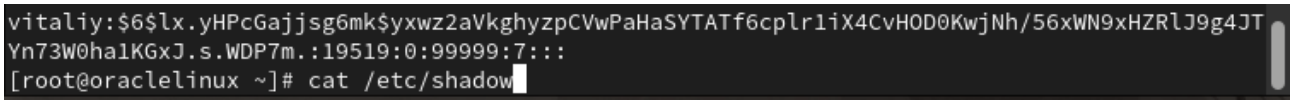

Рисунок 2.6 – Хеш пароля користувача `vitaliy`

У даному випадку використовується алгоритм хешування SHA-512, оскільки хеш починається з 6

Парольна аутентифікація є широко застосовуваним методом у багатьох системах, включаючи операційну систему Linux. Однак, останнім часом все більшу популярність набувають інші методи аутентифікації, такі як ключі в SSH, біометричні дані та інші, які надають додатковий рівень безпеки.

2.2.2 Модуль аутентифікації PAM

PAM є фреймворком аутентифікації у Linux, який дозволяє розширити та налаштувати методи аутентифікації в операційній системі [5]. Використовуючи PAM, можна налаштувати різні способи аутентифікації для різних служб та додатків.

PAM базується на концепції модульності, де кожен аспект аутентифікації (наприклад, перевірка пароля, перевірка біометричних даних, аутентифікація за допомогою токена і т.д.) реалізується як окремий модуль. Коли процес аутентифікації відбувається, PAM викликає ці модулі у визначеному порядку та обробляє їх результати.

Для налаштування PAM використовується конфігураційний файл `/etc/pam.d/`. У цьому каталозі знаходяться окремі файли конфігурації для

різних служб та додатків. Кожен файл конфігурації містить правила, що визначають порядок та параметри виклику модулів PAM для конкретної служби.

Конфігураційні файли PAM мають загальну структуру, яка складається з рядків з ключовими словами та параметрами.

Основні ключові слова включають:

- `auth` – використовується для аутентифікації користувача;
- `account` – використовується для перевірки прав користувача;
- `password` – використовується для зміни пароля користувача;
- `session` – використовується для управління сеансом користувача.

Кожне ключове слово може мати свої параметри, які вказують, які модулі PAM повинні бути викликані та як їх результати обробляються.

Один з важливих аспектів PAM - це можливість налаштування різних методів аутентифікації для різних служб або навіть для різних рівнів безпеки. PAM надає широкий набір модулів аутентифікації, і також можна встановлювати додаткові модулі для розширення функціональності.

Загальний процес аутентифікації за допомогою PAM включає виклик відповідних модулів PAM, які перевіряють введені дані, проводять перевірки безпеки, порівнюють збережені дані аутентифікації та повертають результати успіху або невдачі.

PAM є потужним і гнучким інструментом для налаштування методів аутентифікації в Linux і дозволяє забезпечити безпеку та контроль доступу до системи та її ресурсів.

2.3 Мережева безпека

2.3.1 Фаєрвол та безпека операційної системи

Фаєрвол є одним з найважливіших інструментів для захисту мережі в операційній системі Linux. В Linux доступні різні файрволи, такі як iptables, nftables та firewalld, які надають можливості контролювання трафіку в системі.

Iptables.

Iptables є стандартним файрволом для більшості дистрибутивів Linux [6]. Він оперує на рівні пакетів та працює з таблицями, правилами та ланцюжками.

Правила iptables встановлюються у вигляді ланцюжків (chains). Існують стандартні ланцюжки, такі як INPUT, OUTPUT та FORWARD, але також можна створити свої власні. Можна додавати правила до ланцюжків, щоб визначити, які дії виконувати для певних видів трафіку.

Синтаксис команд iptables включає в себе ряд параметрів та аргументів, які визначають правила для обробки мережевого трафіку. Основні параметри включають наступне:

- `-A` або `–append` - додає правило до кінця вказаного ланцюжка;
- `-D` або `–delete` - видаляє правило з вказаного ланцюжка;
- `-I` або `–insert` - вставляє правило у вказану позицію в ланцюжку;
- `-R` або `–replace` - замінює вказане правило в ланцюжку;
- `-L` або `–list` - виводить список правил в заданому ланцюжку;
- `-F` або `–flush` - видаляє всі правила з вказаного ланцюжка;
- `-P` або `–policy` - встановлює політику за замовчуванням для заданого ланцюжка.

Крім того, для визначення параметрів правила використовуються наступні аргументи:

- `-p` або `–protocol` - вказує протокол трафіку (наприклад, tcp, udp, icmp);
- `-s` або `–source` - вказує IP-адреси або мережі джерела;
- `-d` або `–destination` - вказує IP-адреси або мережі призначення;
- `-i` або `--in-interface` - вказує вхідний мережевий інтерфейс;
- `-o` або `--out-interface` - вказує вихідний мережевий інтерфейс;
- `--sport` - вказує вихідний порт (тільки для TCP або UDP);
- `--dport` - вказує порт призначення (тільки для TCP або UDP);
- `-j` або `–jump` - вказує ціль дії для пакету, якщо він відповідає правилу (наприклад, ACCEPT, DROP, LOG).

В iptables, ланцюжок є послідовністю правил, які застосовуються до мережевого трафіку. Кожен пакет, що проходить через систему, перевіряється протягом цих ланцюжків, і йому може бути надано певну дію в залежності від

визначених правил. В Linux існує кілька стандартних ланцюжків, а також можливість створення власних.

Основні стандартних ланцюжки включають:

- `INPUT` – цей ланцюжок використовується для обробки вхідних пакетів, до системи. Він перевіряє пакети на вхід на мережевих інтерфейсах та вирішує, що робити з ними;
- `FORWARD` – цей ланцюжок використовується для обробки пакетів, які пересилаються через систему. Він перевіряє пакети, що надходять на одному мережевому інтерфейсі та призначені для іншого, і вирішує, чи дозволити або заборонити їх пересилання;
- `OUTPUT` – цей ланцюжок використовується для обробки пакетів, що виходять з системи. Він перевіряє пакети перед відправкою мережеві інтерфейси та вирішує, що робити з ними;
- `PREROUTING` – цей ланцюжок використовується для обробки пакетів до того, як вони пройдуть операцію маршрутизації. Він застосовується до пакетів, які надходять до системи перед тим, як вони будуть маршрутизовані;
- `POSTROUTING` - цей ланцюжок використовується для обробки пакетів після того, як вони пройшли операцію маршрутизації. Він застосовується до пакетів перед їх відправленням з системи.

Кожен ланцюжок складається з правил, які визначають дії, які треба виконати з пакетами, що проходять через них. Правила можуть бути додані, видалені або змінені з використанням різних команд iptables. Синтаксис правил iptables включає специфікацію вхідних та вихідних інтерфейсів, адреси та порти, стани з'єднань і багато іншого.

Наприклад, ось приклад команди для додавання простого правила в ланцюжок INPUT, яке дозволяє вхідний трафік на порт 80 (HTTP):

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

У цьому прикладі:

- `-A INPUT` вказує, що правило додається до ланцюжка INPUT;
- `-p tcp` вказує, що це правило застосовується до TCP-протоколу;

- `--dport 80` вказує, що правило застосовується до пакетів зі значенням призначеного порту 80;

- `-j ACCEPT` вказує, що пакети, що відповідають цьому правилу, мають бути прийняті.

Nftables.

Nftables є новішою системою файрволу у Linux, яка замінює попередню систему iptables. Основною особливістю nftables є його простий та зрозумілий синтаксис, який дозволяє зручно створювати та керувати правилами файрволу [7].

Синтаксис nftables базується на конструкції команд та виразів. Кожна команда виконує певну операцію, таку як створення таблиці, ланцюжка або правила, а вирази використовуються для визначення умов та дій в рамках правил.

Основні елементи синтаксису nftables включають:

1) Таблиці.

Таблиці використовуються для організації логічно пов'язаних правил. Створення таблиці виглядає наступним чином:

```
table <family> <table_name>
```

де:

- `<family>` - вказує сімейство протоколів, з яким пов'язана таблиця. Це може бути одне з значень: `ip, ip6, inet, arp, bridge, netdev`. Вибір сімейства протоколів залежить від типу мережі, з якою ви працюєте;
- `<table_name>` - ім'я таблиці.

Приклад:

```
table ip filter
```

У цьому прикладі:

- `ip` – сімейство протоколів, яке визначає, що таблиця пов'язана з IPv4 протоколом;
- `filter` – назва таблиці.

2) Ланцюжки.

Ланцюжки використовуються для групування правил. Вони визначають, що робити з пакетами, які відповідають певним умовам. Створення ланцюжка виглядає так:

```
chain <chain_name> { type <type> hook <hook> priority <priority>; }
```

де:

- `<chain_name>` - це ім'я ланцюжка;
- `<type>` - тип ланцюжка (`filter`, `nat`, `mangle` тощо);
- `<hook>` - місце вставки ланцюжка. Це може бути одне з значень: `prerouting, input, forward, output, postrouting;`
- `<priority>` - визначає пріоритет ланцюжка в контексті `<hook>`. Менші значення пріоритету виконуються раніше.

Приклад:

```
chain input {
    type filter;
    hook input priority 0;
}
```

У цьому прикладі:

- `input` – назва ланцюжка;
- `filter` – тип ланцюжка, що використовується для фільтрації пакетів;
- `input` – хук, що пов'язаний з вхідним трафіком;
- `priority 0` – пріоритет ланцюжка, вказує, що цей ланцюжок буде виконуватися першим у хуці input.

3) Правила.

Правила використовуються для визначення умов та дій, які потрібно застосувати до пакетів. Вони додаються до ланцюжка за допомогою команди `add rule`. Приклад структури правила:

```
add rule <table_name> <chain_name> <expression> <action>
```

тут:

- `<table_name>` - вказує назву таблиці, до якої буде додано правило;
- `<chain_name>` - вказує назву ланцюжка всередині таблиці, до якого буде додано правило;

- `<expression>` - визначає умови або критерії, які повинен відповідати пакет, щоб відповідати правилу. Вирази можуть включати різні поля, такі як IP-адреси джерела/призначення, порти, протоколи, мітки пакетів та інше. Ці умови використовуються для зіставлення пакетів з правилом;
- `<action>` - вказує дію, яка буде виконана, якщо пакет відповідає правилу. Дії можуть включати `accept` (дозволити пакет), `drop` (відкинути пакет), `reject` (відкинути пакет і надіслати ICMP-відповідь), `jump` (передати управління до іншого ланцюжка), `nat` (виконати NAT), `counter` (підрахунок пакетів та байтів) та інше.

Приклад:

```
add rule filter input ip saddr 192.168.0.0/24 tcp dport 22 counter accept
```

У цьому прикладі:

- `filter` - назва таблиці;
- `input` – назва ланцюжка;
- `ip saddr 192.168.0.0/24 tcp dport 22` - вираз, який вказує, що пакет повинен мати IP-адресу джерела у діапазоні 192.168.0.0/24 та TCP-порт призначення 22 (SSH);
- `counter accept` - дія, яка підраховує пакети та байти та дозволяє відповідні пакети.

4) Вирази.

Вирази в nftables використовуються для визначення умов та фільтрації пакетів у правилах. Вони можуть містити різні компоненти, такі як заголовки протоколів, поля, значення, операції порівняння та логічні оператори. Ось докладніше про деякі компоненти виразів:

5) Заголовки протоколів.

Ви можете використовувати заголовки протоколів, такі як IP, TCP, UDP, ICMP і т.д., для визначення умов фільтрації пакетів. Наприклад, `ip, tcp, udp, icmp, ipv6`, тощо.

Можливо використати конкретні поля заголовків протоколів для порівняння значень. Наприклад, `src, dst, sport, dport, protocol`, тощо.

Також можна вказати конкретні значення для порівняння з полями. Наприклад, IP-адреси, номери портів, числа, рядки тощо.

Ви можете використовувати операції порівняння для перевірки відповідності значень. Наприклад, == (рівність), != (нерівність), < (менше), > (більше), <= (менше або рівне), >= (більше або рівне) тощо.

Ви можете використовувати логічні оператори, такі як && (логічне І), || (логічне АБО), ! (логічне НЕ), для комбінування умов та створення складних виразів.

Наприклад:

```
ip saddr 192.168.0.0/24 tcp dport 22 counter accept
```

У цьому прикладі `ip saddr 192.168.0.0/24` перевіряє, чи адреса джерела пакета знаходиться в мережі 192.168.0.0/24, `tcp dport 22` перевіряє, чи це TCP-пакет з призначеним портом 22, а `counter accept` збільшує лічильник та дозволяє пакет.

6) Дії.

Дії використовуються для визначення того, що робити з пакетом, якщо він відповідає заданим умовам. Ось деякі загальні дії, які можна використовувати в nftables:

- `accept` - приийняти пакет і дозволити його проходження через систему;
- `drop` - відкинути пакет без будь-якого повідомлення або відповіді;
- `reject` - відкинути пакет, але надіслати відповідь про відкидання пакета відправнику;
- `dnat` - змінити адресу та/або порт пакета для перенаправлення його на іншу адресу або порт;
- `snat` - змінити адресу та/або порт пакета для заміни їх на адресу або порт системи;
- `masquerade` - замаскувати (маскувати) адресу вихідного пакета, щоб забезпечити анонімність або відповідність мережевим вимогам;
- `redirect` - перенаправити пакет на інший порт або сокет;
- `counter` - облік пакетів, що відповідають заданим умовам;
- `log` - записати інформацію про пакет в системний журнал.

Firewalld.

Firewalld є інструментом управління файрволом для Linux, який надає зручний інтерфейс для керування правилами безпеки мережі. Він використовуєся в багатьох сучасних дистрибутивах Linux, таких як Fedora, CentOS, Oracle Linux і Red Hat Enterprise Linux.

Основні особливості Firewalld:

1) Зони безпеки (Zone-based Security). Firewalld використовує поняття "зон" для категоризації інтерфейсів мережі. Кожен інтерфейс призначається до певної зони, і для цієї зони встановлюються правила фаєрволу. Це дозволяє легко керувати набором правил для різних зон, наприклад, "зовнішня" зона може мати більш обмежувальні правила, ніж "внутрішня" зона.

2) Поділ сервісів (Service Diversion). Firewalld використовує поняття "сервісів" для групування наборів правил, пов'язаних з конкретними службами або програмами. Замість визначення окремих правил для кожного порту або протоколу, ви можете використовувати вбудовані або власні сервіси, що спрощує керування правилами.

3) Динамічне оновлення (Dynamic Updates). Firewalld може автоматично оновлювати фаєрвол-правила, коли змінюються конфігурація мережі. Це означає, що ви можете додавати або видаляти інтерфейси, змінювати їх зони або сервіси, і фаєрвол буде оновлювати правила без необхідності перезавантаження або втрати з'єднань.

4) Підтримка IPv4 та IPv6. Firewalld повністю підтримує як IPv4, так і IPv6 протоколи, що дозволяє налаштовувати правила фаєрволу для обох типів мереж.

5) Дружній інтерфейс командного рядка та графічний інтерфейс. Firewalld має як командний рядок, так і графічний інтерфейс для керування правилами фаєрволу. Це робить його досить доступним навіть для користувачів без досвіду в управлінні фаєрволом.

Використовуючи один з цих фаєрволів можна налаштувати правила, які керують трафіком в системі. Це дозволяє обмежувати доступ до різних сервісів і

ресурсів, забороняти або дозволяти конкретний мережевий трафік в залежності від потреби. Налаштування фаєрволу в Linux є важливою складовою безпеки.

2.3.2 Безпека мережевого трафіку

У Linux є різні методи шифрування мережевого трафіку, які забезпечують безпеку при передачі даних через мережу [8].

Ось деякі з них.

SSL/TLS.

SSL і його наступник TLS є протоколами, які забезпечують шифрування та ідентифікацію сторін при з'єднанні через мережу. Вони часто використовуються для шифрування HTTP-запитів і відповідей, утворюючи протокол HTTPS. Протоколи SSL/TLS використовують сертифікати для ідентифікації сервера та встановлення захищеного з'єднання.

Основні характеристики SSL/TLS:

- Шифрування. SSL/TLS використовує симетричне та асиметричне шифрування для захисту переданих даних. Симетричне шифрування використовує один ключ для шифрування і розшифрування, тоді як асиметричне шифрування використовує пару ключів - приватний та публічний.
- Аутентифікація. SSL/TLS дозволяє перевірку автентичності сервера і, в деяких випадках, клієнта. Це забезпечує впевненість у тому, що ви спілкуєтеся з вірним сервером і що ваша комунікація захищена.
- Цілісність даних. SSL/TLS використовує хеш-функції для перевірки цілісності переданих даних. Це дозволяє виявляти будь-які зміни або спотворення даних під час передачі.
- Протоколи. SSL/TLS підтримує різні версії протоколів, такі як SSLv3, TLSv1.0, TLSv1.1, TLSv1.2 та TLSv1.3. Кожна версія має свої характеристики та рівень безпеки.

Короткий огляд різних версій SSL/TLS.

SSLv3 використовує симетричне та асиметричне шифрування для захисту передачі даних. Протокол SSLv3 став вразливим до деяких серйозних атак, тому рекомендується уникати його використання і переходити на більш безпечні версії TLS.

TLSv1.0 є наступною версією після SSLv3 і є оновленою версією протоколу. Він включає в себе покращені механізми шифрування та аутентифікації. TLSv1.0 ще використовує застарілі алгоритми шифрування, які можуть бути вразливими до атак. Зараз TLSv1.0 вважається застарілим і не рекомендується для використання.

TLSv1.1 є покращеною версією TLSv1.0 і включає в себе більш сильні алгоритми шифрування та покращені механізми безпеки. Він виправляє деякі вразливості, що були присутні в TLSv1.0. TLSv1.1 все ще використовується, але вважається менш безпечною в порівнянні з більш новими версіями.

TLSv1.2 є широко використовуваною версією протоколу TLS. Він включає в себе сильні алгоритми шифрування та покращені механізми безпеки. TLSv1.2 є значно безпечнішою за TLSv1.0 і TLSv1.1 і зазвичай рекомендується використовувати як мінімальну прийнятну версію.

TLSv1.3 є найновішою версією протоколу TLS, яка пропонує покращену безпеку та продуктивність. Він включає в себе сучасні алгоритми шифрування та механізми безпеки, які зменшують ризик атак і покращують продуктивність з'єднання. TLSv1.3 вважається найбезпечнішою версією протоколу TLS і рекомендується для використання, якщо це можливо.

Важливо зазначити, що рівень безпеки шифрування SSL/TLS також залежить від використаного шифрувального алгоритму та конфігурації сервера та клієнта. Рекомендується використовувати сильні алгоритми шифрування та актуальні версії протоколу TLS для забезпечення максимальної безпеки мережевого зв'язку.

SSL/TLS широко використовується для захисту комунікації на веб-серверах, електронній пошті, месенджерах та інших мережевих протоколах. Він забезпечує безпеку передачі чутливої інформації, такої як паролі, фінансові дані та особисті дані, і забезпечує високий рівень захисту від перехоплення та зламу.

2.4 Шифрування даних

Шифрування даних в Linux використовується для захисту конфіденційності та цілісності інформації, яка передається через мережі або зберігається на системі [9]. Існує кілька методів шифрування даних в Linux, включаючи:

Симетричне шифрування.

У симетричному шифруванні використовується один і той же ключ для шифрування та розшифрування даних. Це швидкий метод шифрування, і найпопулярнішим алгоритмом симетричного шифрування є AES.

Асиметричне шифрування.

У асиметричному шифруванні використовуються два ключі: публічний ключ для шифрування даних і приватний ключ для їх розшифрування. Це дозволяє безпечно обмінюватися шифрованою інформацією між двома або більше сторонами. Розповсюдженими алгоритмами асиметричного шифрування є RSA та ECC.

Хешування.

Хеш-функції використовуються для перетворення даних у фіксований хеш-код. Хеш-код є унікальним для кожного набору даних, і його неможливо відновити до вихідних даних. Хеш-функції широко використовуються для перевірки цілісності даних, а також для збереження паролів у хешованому вигляді, що забезпечує додатковий рівень безпеки.

В Linux існують різні інструменти та бібліотеки для реалізації шифрування даних, включаючи OpenSSL, GnuPG (GPG), libgcrypt та інші. Ці інструменти надають розширені функції шифрування для захисту даних на різних рівнях, включаючи файлове шифрування, шифрування комунікаційних каналів та інше.

2.4.1 Шифрування даних за допомого LUKS

LUKS - це стандарт для шифрування дискових просторів в Linux [10]. Використовуючи LUKS, можна зашифрувати цілий диск або створити зашифрований контейнер для зберігання даних.

Основні особливості LUKS:

Шифрування на рівні блоків.

LUKS шифрує дані на рівні блоків, що дозволяє шифрувати весь диск або окремі розділ.

Підтримка різних шифрувальних алгоритмів. LUKS підтримує різні шифрувальні алгоритми, такі як AES, Twofish, Serpent та інші. Ми можемо вибрати алгоритм, який найкраще відповідає нашим потребам з точки зору безпеки та продуктивності.

Керування ключами. LUKS дозволяє керувати ключами шифрування. Ви можете створювати, змінювати та видаляти ключі шифрування. Кожен ключ може мати свої права доступу і пароль для розблокування диска або контейнера.

Підтримка паролів та ключів. LUKS дозволяє використовувати паролі або ключі для розблокування зашифрованого простору. Паролі можуть бути встановлені для користувачів, а ключі можуть бути збережені в файлі або на USB-накопичувачі.

На рисунках 2.7-2.9 показано процедуру шифрування USB накопичувача за допомогою LUKS. Накопичувач буде зашифровано за допомогою алгоритму AES-XTS-PLAIN64

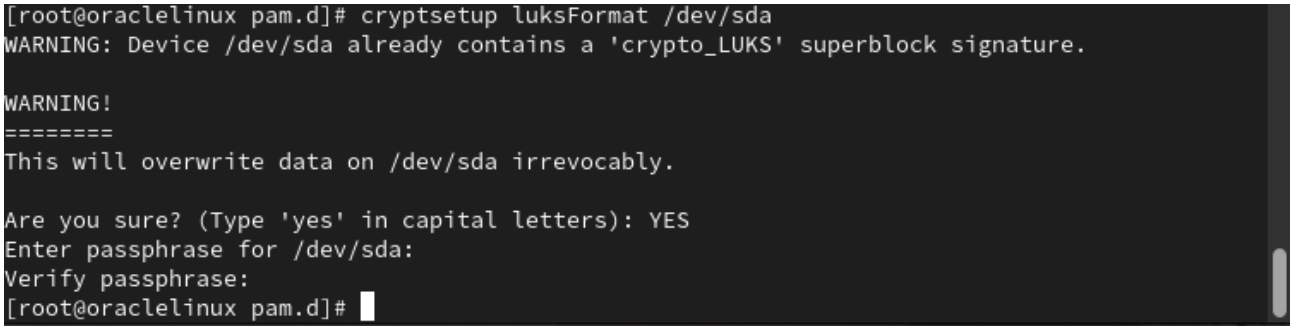

Рисунок 2.7 – Створення нового LUKS-контейнера на USB-накопичувачі

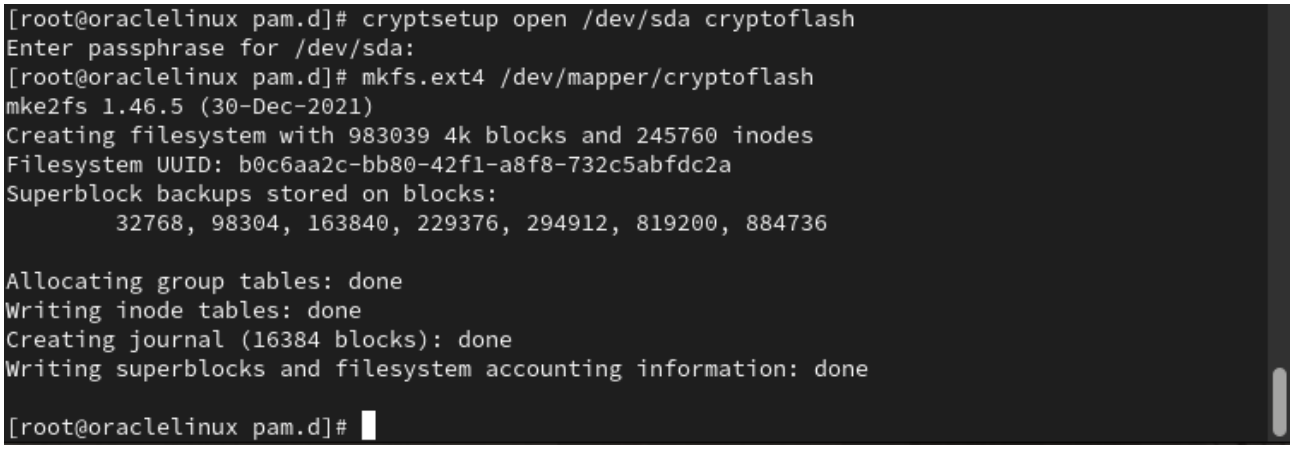

Рисунок 2.8 – Відкриття LUKS-контейнера та створення файлової системи

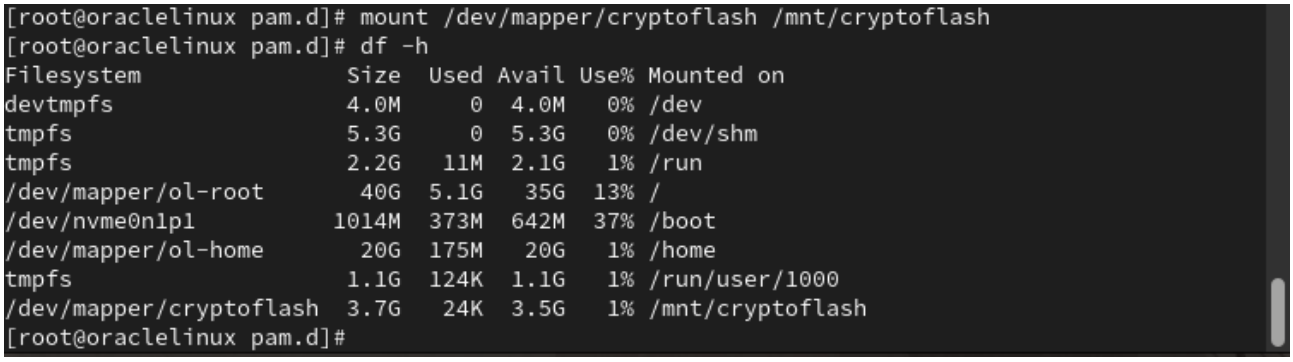

Рисунок 2.9 – Монтування розшифрованої файлової системи

Тепер USB-накопичувач зашифрований за допомогою LUKS і готовий до використання. При кожному підключенні USB-накопичувача буде запропоновано ввести пароль для розшифрування та отримання доступу до даних на пристрої.

За допомогою команди `cryptsetup luksDump /dev/sda` виведемо інформацію про зашифрований диск, таку як ключі, алгоритми шифрування та інше (Рис 2.10).

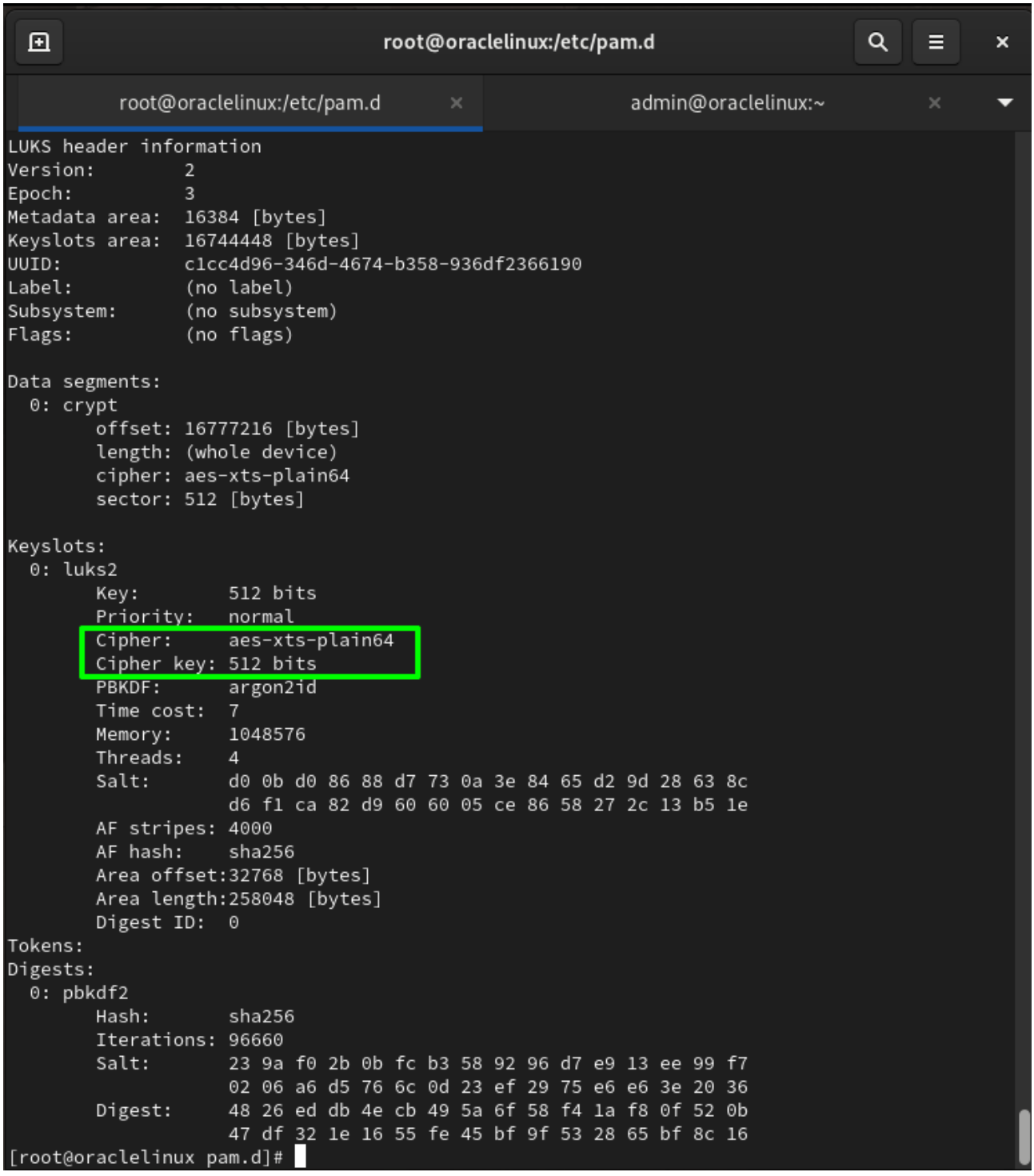

Рисунок 2.10 – Вивід докладної інформації про зашифрований USB-накопичувач

2.4.2 Шифрування даних за допомого EncFS

EncFS - це програма для створення зашифрованих каталогів та файлів у Linux-системах [11]. EncFS працює на рівні файлової системи і надає можливість шифрувати дані в певному каталозі за допомогою ключа шифрування.

Зашифровані дані зберігаються в окремих файлових блоках, які можна монтувати і декодувати.

Основні особливості EncFS.

Легкість використання.

EncFS має простий і зрозумілий інтерфейс, що робить його доступним для широкого кола користувачів. Зашифровані дані виглядають і працюють як звичайні файли і каталоги.

Прозоре шифрування.

EncFS забезпечує прозоре шифрування, що означає, що можна працювати з зашифрованими даними безпосередньо, не помічаючи процесу шифрування або розшифрування. Всі операції шифрування відбуваються автоматично.

Індивідуальне шифрування файлів.

EncFS шифрує кожен файл окремо, що дозволяє зберігати різні файли з різними рівнями шифрування. Це дозволяє збільшити безпеку і забезпечити гнучкість в управлінні шифруванням.

Підтримка криптографічних алгоритмів.

EncFS підтримує різні криптографічні алгоритми шифрування, такі як AES, блочний шифр, який використовується для шифрування даних.

Можливість монтажу на різних системах.

Зашифровані дані, створені за допомогою EncFS, можна монтувати на різних системах, включаючи Linux, FreeBSD і Windows (за допомогою додваткового програмного забезпечення). Це дає нам можливість отримувати доступ до своїх даних з будь-якого пристрою.

Парольний доступ.

EncFS використовує пароль для шифрування та розшифрування даних. Важливо берегти пароль від несанкціонованого доступу, оскільки він є основою для доступу до зашифрованих даних.

Приклад виконання шифрування показано на рисунках 2.11-2.12.

Тека /root/encrypted використовується для зберігання зашифрованих даних а тека /root/dencrypted для монтування розшифрованого тому.

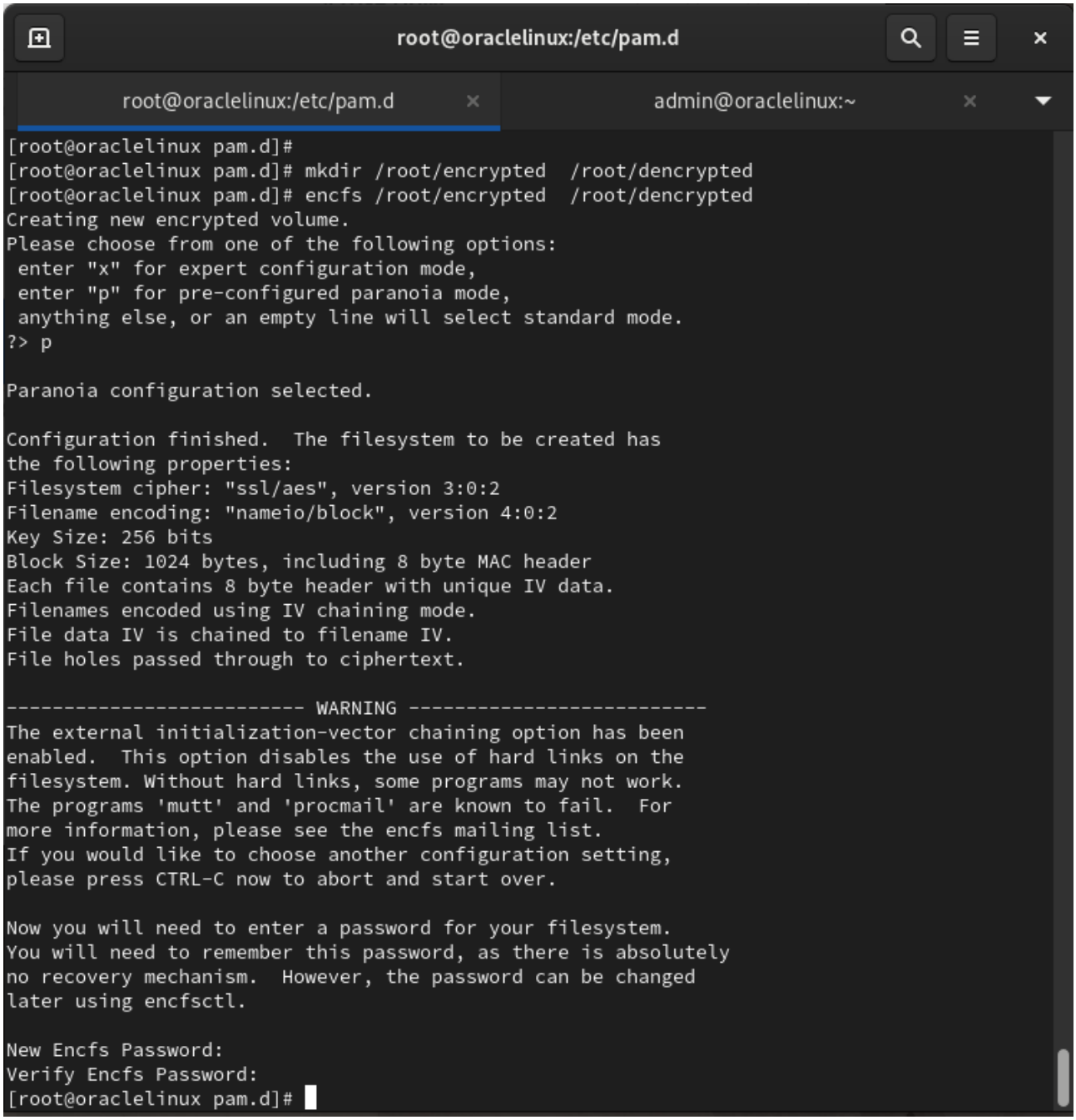

Рисунок 2.11 – Створення віртуального зашифрованого тому

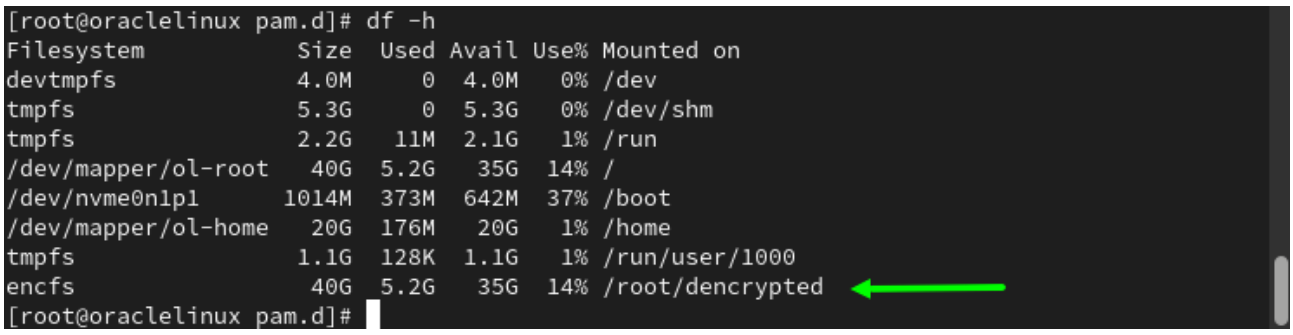

Рисунок 2.12 – Змонтований розшифрований том

Створимо текстовий файл `testencfs.txt` з довільним вмістом в теці /root/dencrypted (Рис.2.13).

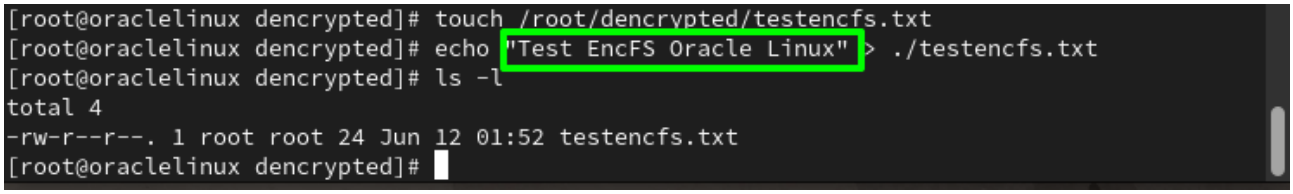

Рисунок 2.13 – Створення текстовий файл `testencfs.txt`

Після створення файлу та збереження його вмісту у теці /root/ dencrypted в теці /root/encrypted буде збережено зашифрований файл (Рис.14).

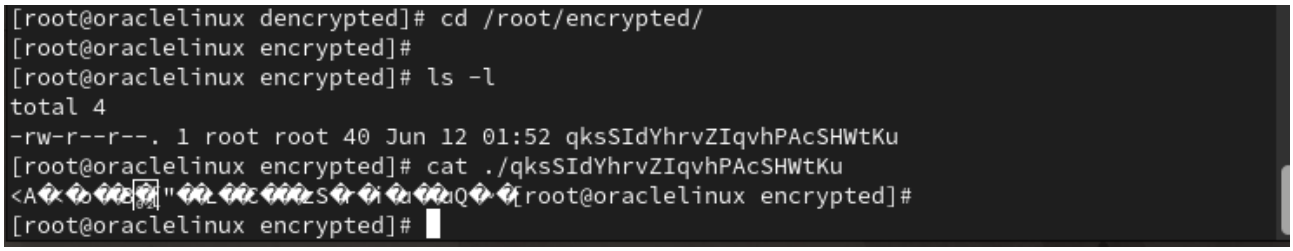

Рисунок 2.14 – Зашифрована назва та вміст файлу `testencfs.txt`

Важливо зазначити, що EncFS є програмою на рівні користувача і не надає криптографічного захисту на рівні ядра. Для більш високого рівня безпеки рекомендується використовувати додатково інше рішення, таке як LUKS, для шифрування цілих пристроїв або розділів.

2.5 Захист від вірусів та шкідливих програм

Захист від вірусів та шкідливих програм є важливим аспектом безпеки в операційній системі Linux.

Основні методи захисту від вірусів і шкідливих програм включають:

1) Встановлення антивірусного програмного забезпечення.

Хоча загрози вірусів для Linux менш поширені порівняно з іншими операційними системами, встановлення антивірусного програмного забезпечення може допомогти виявляти та блокувати потенційні загрози;

2) Оновлення системи.

Регулярне оновлення операційної системи та встановлених оновлень для програм є важливим для закриття вразливостей, які можуть бути використані зловмисниками;

3) Моніторинг системи.

Встановлення системи моніторингу, яка відстежує активність системи та сповіщає про підозрілу або небезпечну діяльність, може допомогти вчасно виявити атаки або ненормальну поведінку програм.

2.5.1 Встановлення та налаштування ClamAV

ClamAV - це відкрите програмне забезпечення, призначене для виявлення і боротьби з вірусами, троянськими програмами, шпигунськими програмами та іншими шкідливими програмами [12]. Він працює на платформі Linux і надає надійний захист від шкідливого програмного забезпечення.

ClamAV володіє потужним механізмом виявлення, який використовує вірусні бази даних і сигнатури для ідентифікації шкідливих програм. Він також включає механізми виявлення незрозумілих файлів та невідомих заголовків, що дозволяє виявляти нові шкідливі програми, які можуть бути непоміченими іншими антивірусними програмами.

ClamAV може бути використаний для сканування файлів та директорій на наявність вірусів, а також для інтеграції з поштовими серверами або файловими серверами для автоматичного виявлення шкідливого програмного забезпечення.

Це потужний інструмент безпеки, який допомагає захистити систему Linux від потенційних загроз і забезпечує надійний рівень безпеки для файлів і даних.

У Oracle Linux для встановлення та налаштування ClamAV використовується пакетний менеджер dnf.

Встановлення ClamAV та всіх необхідних пакетів для його роботи:

```
#dnf -y install clamav-scanner-systemd clamav-server clamav-filesystem clamav-server-systemd clamav-data clamav-update clamav-devel clamav-lib clamav
```

Після встановлення усіх необхідних пакетів ClamAV потрібно виконати наступні команди:

```
# systemctl start  clamav-clamonacc.service
# systemctl enable  clamav-freshclam.service
# systemctl start  clamav-freshclam.service
```

```
# systemctl enable  clamd@.service
# systemctl start  clamd@.service
```

де:

- `systemctl enable clamav-clamonacc.service` - включає автоматичний запуск служби `clamav-clamonacc` під час завантаження системи;
- `systemctl start clamav-clamonacc.service` – ця команда запускає службу `clamav-clamonacc`, яка відповідає за моніторинг активності ClamAV та забезпечення захисту в реальному часі;
- `systemctl enable clamav-freshclam.service` – ця команда включає автоматичний запуск служби `clamav-freshclam` під час завантаження системи. `clamav-freshclam` відповідає за оновлення баз даних ClamAV;
- `systemctl start clamav-freshclam.service` – ця команда запускає службу `clamav-freshclam`, яка завантажує оновлення баз даних ClamAV для виявлення нових загроз;
- `systemctl enable clamd@.service` – ця команда включає автоматичний запуск служби `clamd@.service` під час завантаження системи. `clamd@.service` відповідає за роботу ClamAV-сканера;
- `systemctl start clamd@.service` – ця команда запускає службу `clamd@.service`, яка виконує сканування файлів на наявність вірусів та шкідливих програм.

Перевірити статус відповідних служб можна командою `systemctl status`

На рисунку 2.15 показано статус `clamav-freshclam.service`

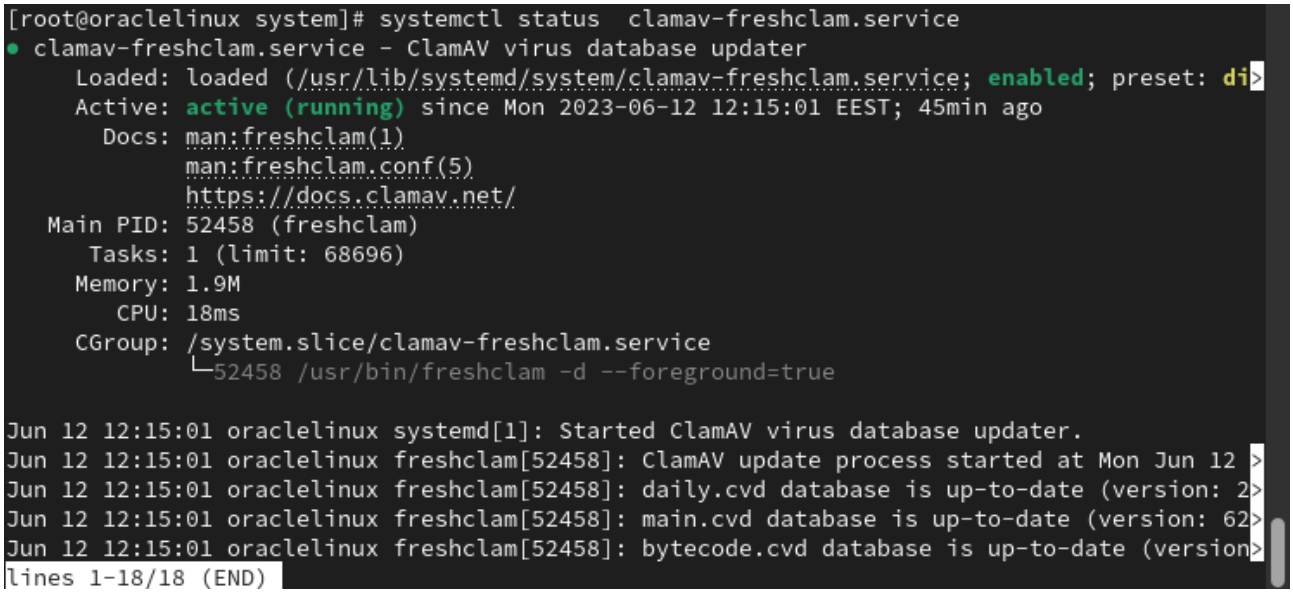

Рисунок 2.15 – Перевірка статусу служби `clamav-freshclam`

2.5.2 Встановлення та налаштування Rootkit Hunter

Rootkit Hunter - це безкоштовний інструмент для виявлення руткітів, шкідливого програмного забезпечення та потенційних загроз безпеки в системі [13]. Він призначений для сканування системних файлів, процесів та скриптів з метою виявлення підозрілих або небезпечних змін.

Основні особливості:

- Виявлення руткітів. Шукає ознаки наявності руткітів, які можуть змінити ядро або скривати свою присутність в системі.
- Сканування системних файлів. Порівнює системні файли зі своєю базою даних підписів, щоб виявити зміни, що можуть вказувати на порушення цілісності.
- Перевірка статусу служб. Перевіряє активні служби та процеси, шукаючи підозрілі або невідомі компоненти.
- Аналіз вразливостей. Перевіряє систему на наявність відомих вразливостей, які можуть бути використані зловмисниками для атаки.
- Звіти та сповіщення. Створює детальні звіти про виявлені загрози та відправляє сповіщення адміністратору системи.

Для встановлення та налаштування Rootkit Hunter в Oracle Linux, слід виконати наступні кроки:

1) Встановлення пакету rkhunter з використанням менеджера пакетів dnf:

```
#dnf install rkhunter
```

2) Після встановлення потрібно виконати команду `rkhunter --update` для оновлення бази даних підписів руткітів:

```
#rkhunter --update
```

3) Налаштування rkhunter проводиться в конфігураційному файлі /etc/rkhunter.conf.

У цьому файлі можна встановити різні параметри, такі як шляхи до директорій для сканування, типи тестів, які слід виконувати, інші налаштування безпеки та звітності.

4) Запуск rkhunter для перевірки системи:

```
#rkhunter --check
```

Rootkit Hunter сканує систему на наявність потенційних загроз, включаючи руткіти, сховані файли, підозрілі процеси та інші аномалії. Зведений результат сканування показано на рисунку 2.16.

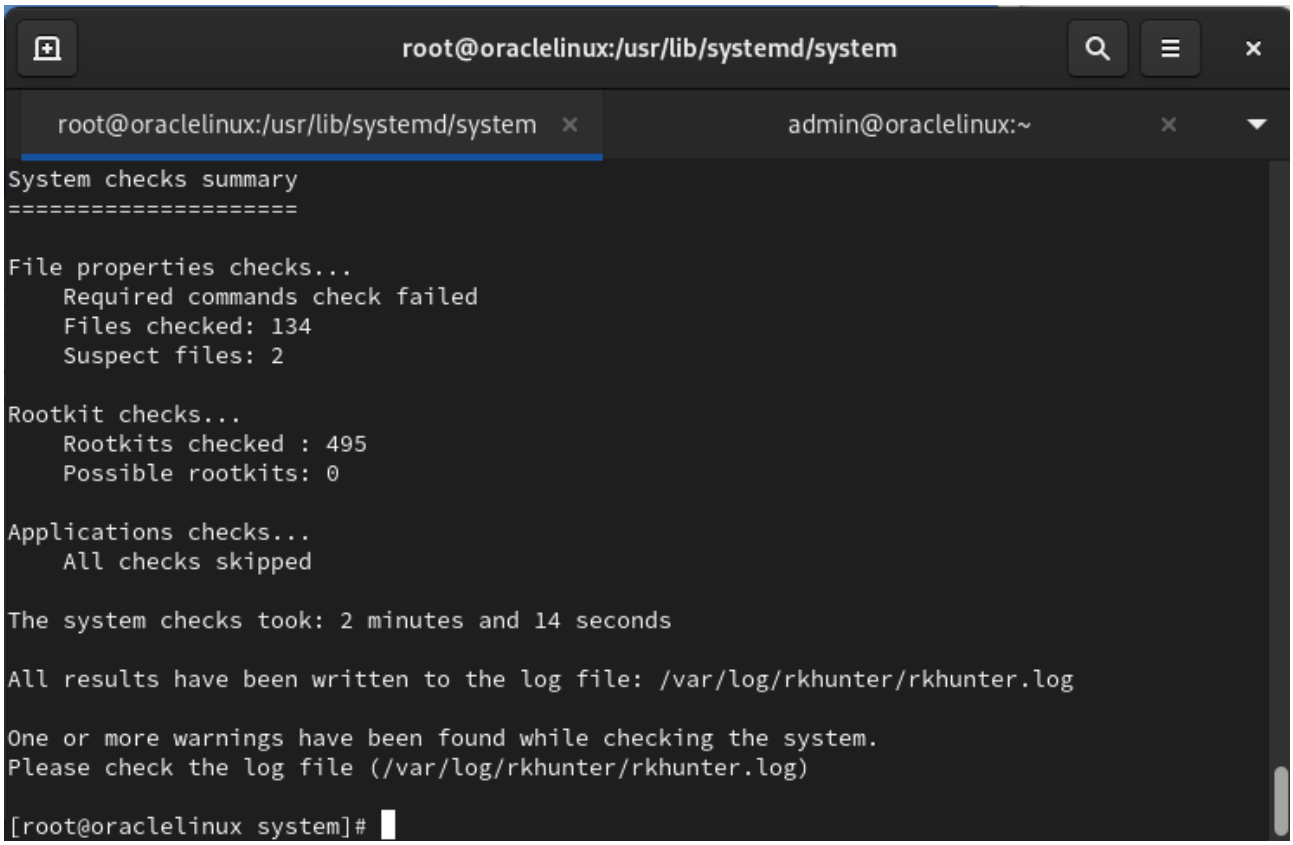

Рисунок 2.16 – Зведений результат сканування `rkhunter --check`

Потрібно періодично запускати оновлення бази даних підписів руткітів, щоб мати актуальну інформацію про відомі загрози. Важливо регулярно оновлювати базу даних підписів та виконувати перевірку системи, щоб виявляти можливі загрози безпеки.

2.6 Моніторинг та аудит подій

Моніторинг та аудит є важливими аспектами забезпечення безпеки в системах Linux. Декілька інструментів, таких як aide та використання лог-файлів дозволяють виконувати моніторинг та аудит системи з метою виявлення неправильних дій, вразливостей та потенційних загроз.

2.6.1 Syslog як механізм логування подій

Syslog є стандартним механізмом логування подій в багатьох операційних системах, включаючи Linux [14]. Він забезпечує спосіб записування системних подій, повідомлень та помилок в централізований журнал для подальшого аналізу та моніторингу.

Основні компоненти syslog включають наступні:

- Служба syslog. Служба syslog є процесом, який слідкує за системними подіями та отримує повідомлення від різних джерел в системі.
- Конфігураційні файли. Для налаштування syslog використовуються конфігураційні файли, такі як /etc/syslog.conf. У цих файлах можна встановити правила, які визначають, які події або повідомлення повинні бути записані в журнал, а також які дії повинні бути виконані з цими повідомленнями (наприклад, збереження в файл, відправка на віддалений сервер тощо).
- Рівні логування. Служба syslog підтримує різні рівні логування, такі як Debug, Info, Warning, Error та Critical. Це дозволяє налаштовувати, які типи повідомлень потрібно записувати в журнал в залежності від важливості.

- Формат повідомлень. Повідомлення, які записуються в журнал, мають певний формат, який зазвичай включає час, джерело події, рівень логування та текстове повідомлення. Цей формат дозволяє зручно аналізувати та фільтрувати журнали.
- Журнали. Інформація про події зберігається в лог-файлах в каталозі /var/log/, де кожен компонент має свій власний лог-файл.

Приклад лог-файлу `/var/log/messages` показано на рисунку 2.17.

```
Jun 12 14:27:19 oraclelinux systemd[1]: Starting dnf makecache...
Jun 12 14:27:20 oraclelinux dnf[148211]: Oracle Linux 9 EPEL Packages for Development (x  1
3 kB/s | 3.0 kB     00:00
Jun 12 14:27:20 oraclelinux dnf[148211]: Oracle Linux 9 BaseOS Latest (x86_64)           2
0 kB/s | 3.6 kB     00:00
Jun 12 14:27:20 oraclelinux dnf[148211]: Oracle Linux 9 Application Stream Packages (x86  1
9 kB/s | 3.9 kB     00:00
Jun 12 14:27:21 oraclelinux dnf[148211]: Oracle Linux 9 UEK Release 7 (x86_64)           1
6 kB/s | 3.0 kB     00:00
Jun 12 14:27:21 oraclelinux dnf[148211]: Metadata cache created.
Jun 12 14:27:21 oraclelinux systemd[1]: dnf-makecache.service: Deactivated successfully.
Jun 12 14:27:21 oraclelinux systemd[1]: Finished dnf makecache.
Jun 12 14:27:21 oraclelinux systemd[1]: dnf-makecache.service: Consumed 1.350s CPU time.
Jun 12 14:28:41 oraclelinux systemd[1]: Starting Fingerprint Authentication Daemon...
Jun 12 14:28:41 oraclelinux systemd[1]: Started Fingerprint Authentication Daemon.
Jun 12 14:28:46 oraclelinux gnome-keyring-daemon[5652]: couldn't initialize slot with maste
r password: The password or PIN is incorrect
Jun 12 14:28:46 oraclelinux NetworkManager[1013]: <info>  [1686569326.6444] agent-manager:
agent[cfc5b6a142f5e656,:1.77/org.gnome.Shell.NetworkAgent/1000]: agent registered
Jun 12 14:29:11 oraclelinux systemd[1]: fprintd.service: Deactivated successfully.

[root@oraclelinux log]# cat /var/log/messages
```

Рисунок 2.17 – Приклад лог-файлу `/var/log/messages`

Файл `/var/log/messages` є одним з основних системних журналів в Linux-подібних операційних системах. В цьому файлі зберігаються різноманітні системні повідомлення, сповіщення та події.

Переваги використання syslog для логування подій включають централізацію логів з різних джерел, можливість фільтрації та аналізу журналів, а також спрощення виявлення проблем та моніторингу системи. Додатково, syslog дозволяє налаштовувати відправку логів на віддалені сервери для зберігання та аналізу на центральній системі моніторингу.

Можна використовувати інструменти, такі як grep, awk або спеціалізовані програми для аналізу та фільтрації логів. Деякі інструменти, такі як ELK Stack (Elasticsearch, Logstash, Kibana), надають розширені можливості для централізованого збору, аналізу та візуалізації лог-даних з різних джерел.

2.6.2 Інструмент моніторингу цілісності файлів AIDE

AIDE є інструментом для моніторингу цілісності файлів у системі [15]. Основна функція AIDE полягає у створенні базового образу системних файлів, який включає хеш-суми файлів і метадані. Після створення базового образу AIDE може періодично перевіряти цілісність файлів і порівнювати їх зі збереженими хеш-сумами.

Основні особливості AIDE:

- Створення базового образу. AIDE починає роботу зі створення базового образу системних файлів. Цей образ включає хеш-суми файлів, атрибути та іншу інформацію про файли, яку ви обираєте;
- Перевірка цілісності. Після створення базового образу AIDE може виконувати періодичну перевірку цілісності файлів. Він порівнює актуальні хеш-суми файлів зі збереженими хеш-сумами в базовому образі і виявляє будь-які зміни, які відбулися;
- Звіти про зміни. Якщо AIDE виявляє зміни у файловій системі, він генерує звіт, що містить детальну інформацію про змінені файли. Це може включати назву файлу, тип зміни, нову хеш-суму та іншу відповідну інформацію;
- Контроль конфігураційних файлів. AIDE також може перевіряти цілісність конфігураційних файлів у системі, забезпечуючи контроль за можливими змінами в конфігурації системи;
- Регулярні перевірки. AIDE може бути налаштований на автоматичну перевірку цілісності файлів з використанням планувальника завдань або іншого механізму планування, що дозволяє регулярно перевіряти систему на зміни.

Приклад встановлення, налаштування та перевірки AIDE в Oracle Linux:

1) Встановлення AIDE:

```
# dnf install aide
```

2) Ініціалізація базового образу AIDE:

```
# aide --init
```

Ця команда створить базовий образ системи, включаючи хеш-суми файлів та іншу метаінформацію. Файл базового образу зберігається у `/var/lib/aide/aide.db.new.gz`.

3) Переміщення базового образу у потрібне місце:

```
# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

4) Налаштування перевірок AIDE:

Редагуючи файл конфігурації AIDE `/etc/aide/aide.conf` за допомогою текстового редактора можна налаштувати правила перевірок згідно своїх потреб.

5) Змінимо пароль користувача `limituser` (Рис.2.18). Це приведе до змін в файлі `/etc/shadow`

```
[root@oraclelinux log]#
[root@oraclelinux log]# passwd limituser
Changing password for user limituser.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

Рисунок 2.18 – Зміна паролю користувача `limituser`

6) Запуск перевірки цілісності файлів:

```
# aide --check
```

Ця команда перевіряє цілісність файлів, порівнюючи їх з базовим образом. Вона виводить звіт про будь-які зміни, виявлені в системі.

На рисунку 2.19 показано результат перевірки та виявлення змін в файлі `/etc/shadow`

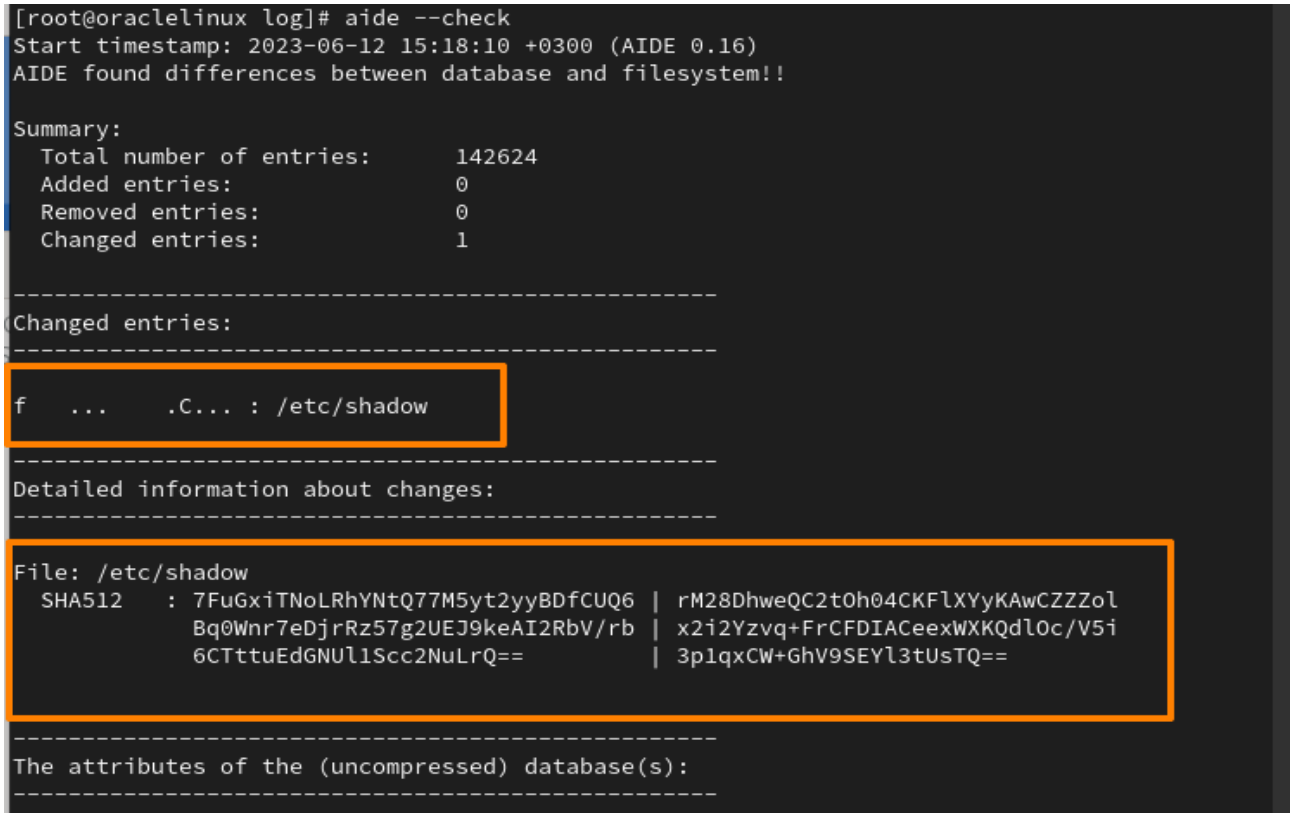

Рисунок 2.19 – Результат перевірки та виявлення змін в файлі `/etc/shadow`

7) Регулярна перевірка цілісності:

Додамо завдання у планувальник завдань (cron) для автоматичного виконання перевірок AIDE з певною регулярністю. Оптимально перевіряти AIDE щодня. Наприклад, щоб запланувати щоденне виконання AIDE о 03:10 за допомогою cron команди, потрібно додати такий рядок до /etc/crontab файлу:

*10 3 * * * root /usr/sbin/aide --check*

AIDE є корисним інструментом для виявлення зловживань, змін у системі та потенційних проблем безпеки. Він дозволяє адміністраторам системи вчасно реагувати на зміни та забезпечувати цілісність системних файлів.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ БЕЗПЕКОВИХ МЕХАНІЗМІВ В LINUX

3.1 Налаштування сервера SSH

3.1.1 Огляд протоколу SSH

SSH (Secure Shell) - це протокол мережевої комунікації, який забезпечує захищене з'єднання між двома вузлами через незахищену мережу [16]. В основному використовується для забезпечення безпечного та захищеного віддаленого доступу до системи, а також для виконання команд на віддалених системах.

Основні особливості SSH.

Шифрування даних.

SSH використовує шифрування для захисту конфіденційності даних, які передаються між клієнтом і сервером. Це означає, що навіть якщо дані перехоплюються на мережевому рівні, вони залишаються незрозумілими для зловмисників.

Аутентифікація.

SSH використовує різні методи аутентифікації для перевірки ідентичності користувача перед наданням доступу. Зазвичай використовуються паролі або ключі SSH.

Перенаправлення портів.

SSH дозволяє перенаправляти мережеві порти між локальною і віддаленою системою. Це дає можливість створювати захищені тунелі для передачі даних між системами, що знаходяться в різних мережах.

Управління файлами.

SSH дозволяє виконувати операції з файлами на віддаленій системі, такі як копіювання, переміщення, видалення тощо. Це дозволяє адміністраторам здійснювати віддалене керування файлами безпосередньо з їх робочої станції.

Керування віддаленими системами.

SSH надає можливість виконувати команди на віддалених системах з локальної системи. Це дозволяє адміністраторам керувати віддаленими серверами та виконувати необхідні дії.

Конфігураційні файли.

SSH має набір конфігураційних файлів, які дозволяють налаштовувати різні аспекти його роботи. Наприклад, можна налаштувати аутентифікацію, шифрування, перенаправлення портів та інші параметри за допомогою цих файлів.

Ключі SSH.

Одним з основних аспектів SSH є використання ключів SSH для аутентифікації. Кожен користувач може створити свої ключі SSH, які складаються з публічного та приватного ключів. Публічний ключ зберігається на сервері, а приватний ключ зберігається в клієнта.

3.1.2 Базові налаштування SSH-сервера

Головний конфігураційний файл SSH sshd_config розташований у директорії /etc/ssh/.

Файл sshd_config містить налаштування SSH-сервера, які визначають його поведінку і параметри безпеки [16].

Основні налаштування sshd_config включають:

- `Port 22` – визначає номер порту, на якому SSH-сервер слухає з'єднання;
- `-PermitRootLogin no` – вказує, чи дозволено аутентифікацію в якості користувача root;
- `PasswordAuthentication yes` – вказує, чи дозволено аутентифікацію за допомогою пароля;
- `AuthorizedKeysFile .ssh/authorized_keys` – вказує шлях до файлу, в якому зберігаються публічні ключі користувачів для аутентифікації;
- `PubkeyAuthentication yes` - вказує, чи дозволено аутентифікацію за допомогою публічних ключів;

- `PermitEmptyPasswords no` – вказує, чи дозволено пусті паролі. За замовчуванням;
- `LogLevel INFO` - вказує рівень журналювання подій SSH;
- `AllowUsers *@192.168.0.*` - ця конфігурація в sshd_config встановлює дохвіл на доступ до SSH-сервера лише для користувачів, які мають IP-адресу з підмережі 192.168.0.0/24.

За допомогою параметра `AllowUsers`, вказаного в такому форматі *@IP-адреса, можна вказати шаблон для обмеження доступу. У цьому конкретному прикладі `*@192.168.0.*`, `*` вказує на будь-яке ім'я користувача, а `192.168.0.*` вказує на будь-яку IP-адресу з підмережі `192.168.0.0/24`.

3.1.3 Алгоритми шифрування та хешування SSH

SSH використовує різні криптографічні шифри та механізми хешування для забезпечення конфіденційності, цілісності та автентифікації даних під час з'єднання між клієнтом і сервером [16]. Шифри використовуються для шифрування трафіку SSH і захисту від атак, таких як перехоплення та модифікація даних.

Розглянемо приклад того, які шифри та механізми хешування можуть використовуватись на кожному етапі встановлення з'єднання SSH:

1) Встановлення з'єднання

Шифри не використовуються на цьому етапі.

2) Протокол транспорту SSH.

Тут обмін параметрами не включає шифрування, але використовується криптографічний хеш-алгоритм, наприклад, SHA-2 (SHA-256, SHA-512), для генерації хешів для перевірки цілісності.

3) Вибір алгоритмів шифрування.

Клієнт і сервер обмінюються списками підтримуваних алгоритмів шифрування, хешування та інших криптографічних алгоритмів.

Наприклад:

Алгоритми шифрування: AES-128-CTR, AES-192-CTR, AES-256-CTR, AES-128-GCM, AES-256-GCM.

Алгоритми хешування: SHA-256, SHA-512.

Алгоритми аутентифікації: RSA.

4) Угода про секретний ключ.

На цьому етапі використовуються алгоритми, такі як Diffie-Hellman (DH) або Elliptic Curve Diffie-Hellman (ECDH), для угоди про спільний секретний ключ. Застосовуються шифрування та хешування для захисту обміну ключами і підтвердження автентичності.

5) Шифрування та з'єднання.

Після угоди про секретний ключ встановлюється шифрована сесія SSH.

На цьому етапі використовуються шифри, такі як AES (наприклад, AES-128, AES-256), для шифрування трафіку між клієнтом і сервером.

6) Аутентифікація.

Варіанти аутентифікації включають в себе пароль або публічний ключ. Шифрування використовується для захисту аутентифікаційних даних. Для захисту аутентифікаційних даних можуть використовуватись різні шифри, такі як RSA або ECC.

3.1.4 Налаштування SSH аутентифікації за допомогою ключів

SSH аутентифікація за допомогою ключів є безпечним методом аутентифікації, який використовує пару криптографічних ключів: приватний ключ і публічний ключ [16]. Приватний ключ зберігається на клієнтській стороні, а публічний ключ на сервері, до якого користувач намагається отримати доступ.

Цей метод забезпечує високий рівень безпеки під час SSH-з'єднань, зменшуючи ризик підміни або перехоплення пароля.

Налаштування SSH аутентифікації за допомогою ключів включає кілька кроків.

Докладний опис процесу:

1) Генерація ключів:

На клієнтській машині з ОС Windows 10 (комп'ютері, з якого ми будемо здійснювати SSH-з'єднання) використаємо PuTTY для генерації ключів SSH.

Відкриємо термінал з PowerShell та введем команду `ssh-keygen -t rsa` для генерації ключів RSA. Можна використовувати інші типи ключів, такі як DSA або ECDSA, але RSA є найбільш поширеним (Рис.3.1).

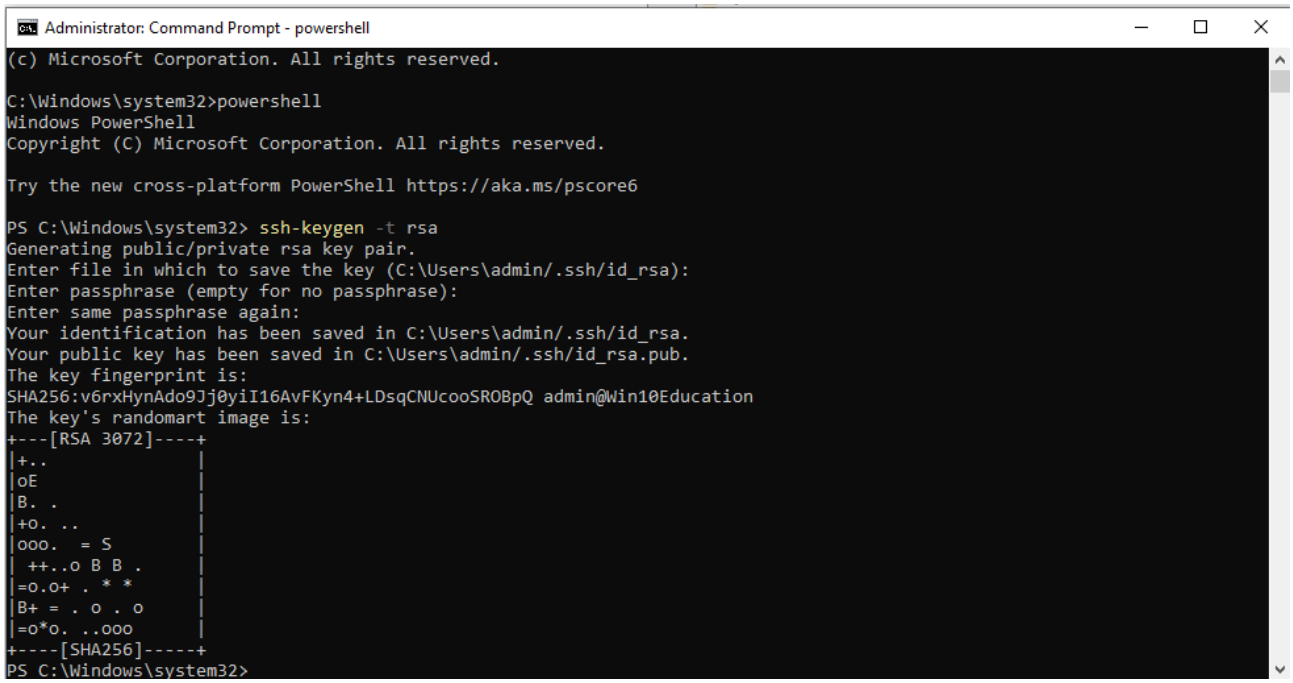

Рисунок 3.1 - Генерації ключів RSA

Система попросить вказати розташування для збереження ключа та запитає про парольну фразу. Можна вказати розташування за замовчуванням та парольну фразу для додаткового захисту ключа, але це необов'язково.

2) Копіювання публічного ключа на сервер:

За допомогою команд в PowerShell здійснимо копіювання публічного ключа на SSH сервер (Рис.3.2)

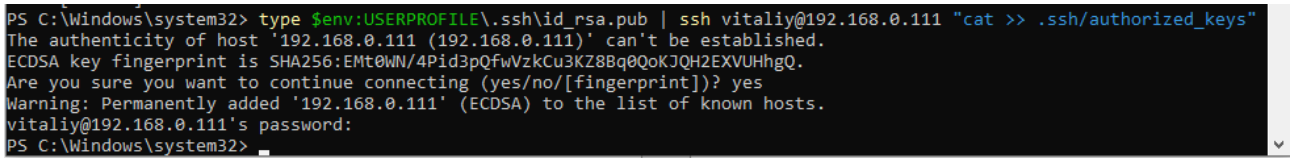

Рисунок 3.2 - Копіювання публічного ключа на SSH сервер

Команда PowerShell використовується для читання вмісту файлу публічного `id_rsa.pub` і надсилання його на віддалений сервер за допомогою SSH для додавання до файлу `authorized_keys`.

Зміст команди наступний:

- `type $env:USERPROFILE\.ssh\id_rsa.pub` - ця частина команди вказує PowerShell прочитати вміст файлу `id_rsa.pub`. `$env:USERPROFILE` є змінною оточення, що містить шлях до домашньої папки користувача в системі Windows;
- `|` - цей символ "pipe" використовується для передачі виводу попередньої команди як вхідних даних для наступної команди;
- `ssh vitaliy@192.168.0.111` - ця частина команди встановлює SSH-з'єднання з віддаленим сервером з IP-адресою 192.168.0.111 з логіном `vitaliy`;
- `"cat >> .ssh/authorized_keys"` - ця частина команди передає команду `cat >> .ssh/authorized_keys` на віддалений сервер через SSH. Команда `cat` виводить вміст файлу, а `>>` вказує, що вміст файлу має бути доданий до кінця файлу `authorized_keys` у папці `.ssh` на віддаленому сервері.

Отже, виконуючи цю команду, ми передаємо вміст файлу `id_rsa.pub` з нашого локального комп'ютера на віддалений сервер та додаємо його до файлу `authorized_keys`, щоб дозволити SSH-аутентифікацію з використанням вказаного ключа.

3) Аутентифікація за допомогою ключа:

Після успішного копіювання ключа ви можете здійснювати SSH-підключення до сервера без введення пароля.

За допомогою команд `ssh -i C:\Users\admin/.ssh/id_rsa vitaliy@192.168.0.111` в PowerShell здійснимо підключення до SSH сервера з використанням приватного ключа (Рис.3.3).

```
PS C:\Windows\system32> ssh -i C:\Users\admin/.ssh/id_rsa vitaliy@192.168.0.111
Last failed login: Mon Jun 12 20:38:43 EEST 2023 from 192.168.0.110 on ssh:notty
There were 2 failed login attempts since the last successful login.
Last login: Mon Jun 12 20:38:28 2023
[vitaliy@oraclelinux ~]$ uname -a
Linux oraclelinux 5.15.0-101.103.2.1.el9uek.x86_64 #2 SMP Tue May 2 01:10:45 PDT 2023 x86_64 x86_64 x86_64 GNU/Linux
[vitaliy@oraclelinux ~]$
```

Рисунок 3.3 - Підключення до SSH сервера

Зміст команди наступний:

- `ssh` – це команда SSH для встановлення з'єднання з віддаленим сервером;

- `-i C:\Users\admin/.ssh/id_rsa` - ця опція вказує шлях до приватного ключа (`id_rsa`), який буде використовуватися для аутентифікації замість стандартного парольного входу;

- `vitaliy@192.168.0.111` - це комбінація імені користувача (`vitaliy`) та IP-адреси віддаленого сервера (`192.168.0.111`), до якого ми намагаємось підключитися.

Ця команда виконує з'єднання з віддаленим сервером за допомогою SSH, використовуючи приватний ключ `id_rsa` для аутентифікації, і вказує ім'я користувача `vitaliy` та IP-адресу віддаленого сервера `192.168.0.111`.

Це загальний опис процесу налаштування SSH аутентифікації за допомогою ключів. Варто відзначити, що процедура може дещо відрізнятися залежно від операційної системи та конкретних налаштувань сервера SSH.

3.2 Налаштування фаєрвола nftables

3.2.1 Базові налаштування фаєрвола nftables

Nftables - це інструмент для фільтрації пакетів в ядрі Linux. Він є наступником і покращеною версією iptables і має більш сучасний синтаксис та функціональність [7]. Основні поняття що до nftables були описані в п.2.3.1.

Для встановлення та налаштування базових правил фаєрвола nftables виконаємо наступні кроки:

1) Створимо файл `base.nft` з базовими налаштуваннями nftables:

```
flush ruleset
table inet base_filter {
 set allowed_protocols {
      type inet_proto
      elements = { icmp, icmpv6 }
 }
 set allowed_interfaces {
      type ifname
      elements = { "lo" }
```

```
    }
    set allowed_tcp_dports {
        type inet_service
        elements = { ssh, 9090, 80, 443 }
    }
    chain allow {
        ct state established,related accept
        meta l4proto @allowed_protocols accept
        iifname @allowed_interfaces accept
        tcp dport @allowed_tcp_dports accept
    }
    chain input {
        type filter hook input priority filter + 20
        policy drop
        jump allow
        reject with icmpx type port-unreachable
    }
}
```

2) Застосуємо дані правила:

```
#nft -f base.nft
```

3) Перевірка працездатності:

Щоб перевірити працездатність, встановимо з'єднання через SSH (на порт 22) до нашого сервера з іншого пристрою з операційною системою Ubuntu Linux. На рисунку 3.4 можна побачити що правила дозволяють підключення по SSH.

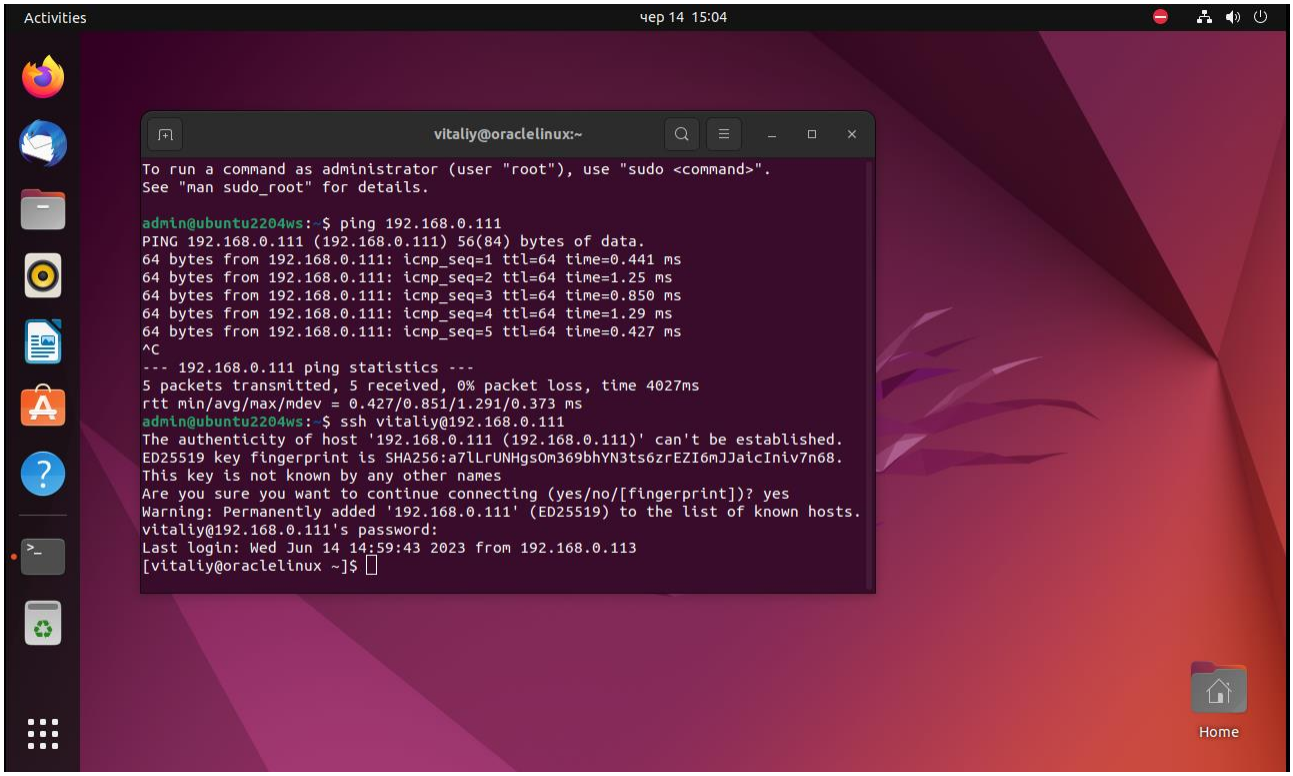

Рисунок 3.4 - Перевірка працездатності базових правил nftables

3.2.2 Динамічне обмеження кількості одночасних з’єднань

Обмеження кількості одночасних з'єднань за допомогою nftable є однією зі стратегій контролю мережевого трафіку, яка дозволяє обмежити кількість одночасних з'єднань, які можуть бути встановлені між двома вузлами.

Ось деякі можливі застосування обмеження кількості одночасних з'єднань за допомогою nftable:

1) Зменшення шкоди від DDoS-атак. Обмеження кількості одночасних з'єднань може допомогти у зменшення шкоди від атак DDoS, де зловмисники намагаються перевантажити сервер, запустивши велику кількість одночасних з'єднань. Застосування обмежень допомагає виявити та блокувати такі атаки, зменшуючи вплив на нормальну роботу мережі;

2) Керування трафіком. Обмеження кількості одночасних з'єднань також може бути використано для керування трафіком в мережі. Наприклад, це може бути корисно в ситуаціях, коли потрібно обмежити кількість одночасних підключень до конкретного ресурсу або сервісу, щоб зберегти пропускну здатність мережі та забезпечити рівномірний доступ до ресурсів;

3) Захисту від brute force атак. Обмеження кількості одночасних з'єднань за допомогою nftable може бути використане для захисту від швидких brute force атак. Брутфорс-атаки - це спроби зламати пароль, перебираючи всі можливі комбінації паролів до вдалого входу.

Розробимо правила nftables, які дозволять зменшити шкоду від DDoS-атак на WEB-сервер та захистять від швидких brute force атаки на SSH-сервер.

1) Створимо файл `limit_filter.nft` з потрібними налаштуваннями nftables:

```
flush ruleset
table inet limit_conn_filter  {
 set allowed_protocols {
      type inet_proto
      elements = { icmp, icmpv6 }
 }
 set allowed_interfaces {
      type ifname
      elements = { "lo" }
 }
 set allowed_tcp_dports {
      type inet_service
      elements = { ssh, 9090, 80, 443 }
 }

 chain allow {
      ct state established,related accept
      meta l4proto @allowed_protocols accept
      iifname @allowed_interfaces accept
      tcp dport @allowed_tcp_dports accept
 }
 chain input {
      type filter hook input priority filter + 20
      policy drop
      jump allow
      reject with icmpx type port-unreachable
 }
```

```
    chain ssh_chain  {
        type filter hook input priority filter+10;
        tcp dport { ssh } ct state new,untracked meter sshmeter {
ip saddr timeout 600m limit rate over 2/minute } reject
    }
    chain web_limit {
        type filter hook input priority filter+10;
        tcp dport { http, https } ct state new,untracked meter
ratemeter  { ip saddr timeout 50m limit rate over 100/minute } drop

    }

}
```

Налаштування в файлі `limit_filter.nft` мають наступні значення:

- `flush ruleset` – використовується для очищення всіх правил в поточному наборі правил nftable. Це означає, що всі правила, які були встановлені раніше, будуть видалені, і набір правил буде порожнім;
- `set allowed_protocols` – вказує, що ми створюємо множину з назвою "allowed_protocols". Ця множина буде містити дозволені протоколи;
- `type inet_proto` – Визначає тип елементів у множині "allowed_protocols" як "inet_proto". Це означає, що в цій множині можуть бути елементи, які представляють протоколи інтернету;
- `elements = { icmp, icmpv6 }` – Визначає елементи, які містяться у множині "allowed_protocols". У цьому випадку, множина містить протоколи ICMP та ICMPv6;
- `set allowed_interfaces` – вказує, що ми створюємо множину з назвою "allowed_interfaces". Ця множина буде містити дозволені інтерфейси;
- `type ifname` – визначає тип елементів у множині "allowed_interfaces" як "ifname". Це означає, що в цій множині можуть бути елементи, які представляють назви інтерфейсів;
- `elements = { "lo" }` – взначає елементи, які містяться у множині "allowed_interfaces". У цьому випадку, множина містить один елемент - "lo", що представляє локальний інтерфейс (Loopback);

- `set allowed_tcp_dports` – вказує, що ми створюємо множину з назвою "allowed_tcp_dports". Ця множина буде містити дозволені TCP порти;
- `type inet_service` – визначає тип елементів у множині "allowed_tcp_dports" як "inet_service". Це означає, що в цій множині можуть бути елементи, які представляють TCP порти;
- `elements = { ssh, 9090, 80, 443 }` – визначає елементи, які містяться у множині "allowed_tcp_dports". У цьому випадку, множина містить чотири елементи, що представляють відповідно порти 22, 9090, 80 і 443.

Ланцюжок `allow`:

- `ct state established,related accept` – це правило перевіряє стан з'єднання пакета. Якщо пакет належить до встановленого або пов'язаного з'єднання, воно буде прийнято (accept). Це дозволяє пропускати пакети, що відносяться до вже встановлених або пов'язаних з'єднань;
- `meta l4proto @allowed_protocols accept` – це правило перевіряє протокол IPv4 (наприклад, TCP або UDP) пакета. Воно перевіряє, чи належить протокол пакета до множини "allowed_protocols", яку ми визначили раніше. Якщо так, то пакет буде прийнято (accept);
- `iifname @allowed_interfaces accept` - це правило перевіряє вхідний інтерфейс пакета. Воно перевіряє, чи належить вхідний інтерфейс пакета до множини "allowed_interfaces", яку ми визначили раніше. Якщо так, то пакет буде прийнято (accept);
- `tcp dport @allowed_tcp_dports accept` – це правило перевіряє TCP-порт призначення пакета. Воно перевіряє, чи належить TCP-порт призначення пакета до множини "allowed_tcp_dports", яку ми визначили раніше. Якщо так, то пакет буде прийнято (accept).

Ланцюжок `input`:

- `type filter hook input priority filter + 20` – це вказує тип ланцюжка правил як "filter" та його точку прикріплення до точки входу пакетів. Параметр "priority" встановлює пріоритет виконання правил у ланцюжку. У цьому випадку, пріоритет встановлено як "filter + 20", що

означає, що цей ланцюжок буде виконуватись після ланцюжка з пріоритетом "filter";

- `policy drop` – це встановлює політику для пакетів, які не співпадають з жодним правилом в ланцюжку. У цьому випадку, політика встановлена як "drop", що означає, що будь-який пакет, який не відповідає жодному правилу, буде відкинутий;
- `jump allow` – це правило пересилає пакети до ланцюжка "allow". Це означає, що пакети, які досягають цього правила, будуть перенаправлені до виконання правил у ланцюжку "allow";
- `reject with icmpx type port-unreachable` – це правило відкидає пакети та генерує відповідь ICMP типу "port-unreachable". Якщо пакет досягає цього правила, він буде відкинутий, а відправнику буде надіслано повідомлення про недоступність порту за допомогою протоколу ICMP.

Ланцюжок `ssh_chain` :

- `type filter hook input priority filter+10` – це вказує тип ланцюжка правил як "filter" та його точку прикріплення до точки входу пакетів. Параметр "priority" встановлює пріоритет виконання правил у ланцюжку. У цьому випадку, пріоритет встановлений як "filter+10", що означає, що цей ланцюжок буде виконуватись після ланцюжка з пріоритетом "filter" і до ланцюжка з пріоритетом "filter+20";
- `tcp dport { ssh }` – це правило перевіряє TCP-порт призначення пакета і співставляє його з портом 22. Якщо порт призначення пакета відповідає порту SSH, то правило співпадає;
- `ct state new,untracked` – це правило перевіряє стан з'єднання. Воно перевіряє, чи стан з'єднання є "new" (нове) або "untracked" (незареєстроване). Якщо стан відповідає одному з цих станів, то правило співпадає;
- `meter sshmeter { ip saddr timeout 600m limit rate over 2/minute }` – це визначає обмеження на кількість підключень по SSH. Застосовується метр "sshmeter", який обмежує кілкість підключень з

однієї IP-адреси джерела. Обмеження встановлено не більше 2 підключень за хвилину;

- `timeout 600m` – цей параметр встановлює таймаут для записів метра. У цьому випадку, таймаут встановлено на 600 хвилин (або 10 годин). Це означає, що після 10 годин записи метра будуть видалені;
- `reject` – це правило відкидає пакети, які відповідають правилу.

2) Застосуємо дані правила:

```
#nft -f limit_filter.nft
```

3) Перевірка працездатності:

Щоб перевірити працездатність, встановимо з'єднання через SSH (на порт 22) до нашого сервера з іншого пристрою з операційною системою Ubuntu Linux. На рисунку 3.5 можна побачити що правила дозволяють підключення по SSH, але не більше двох одночасно, якщо буде перевищено ліміт то IP адреса буде заблокована на 10 годин.

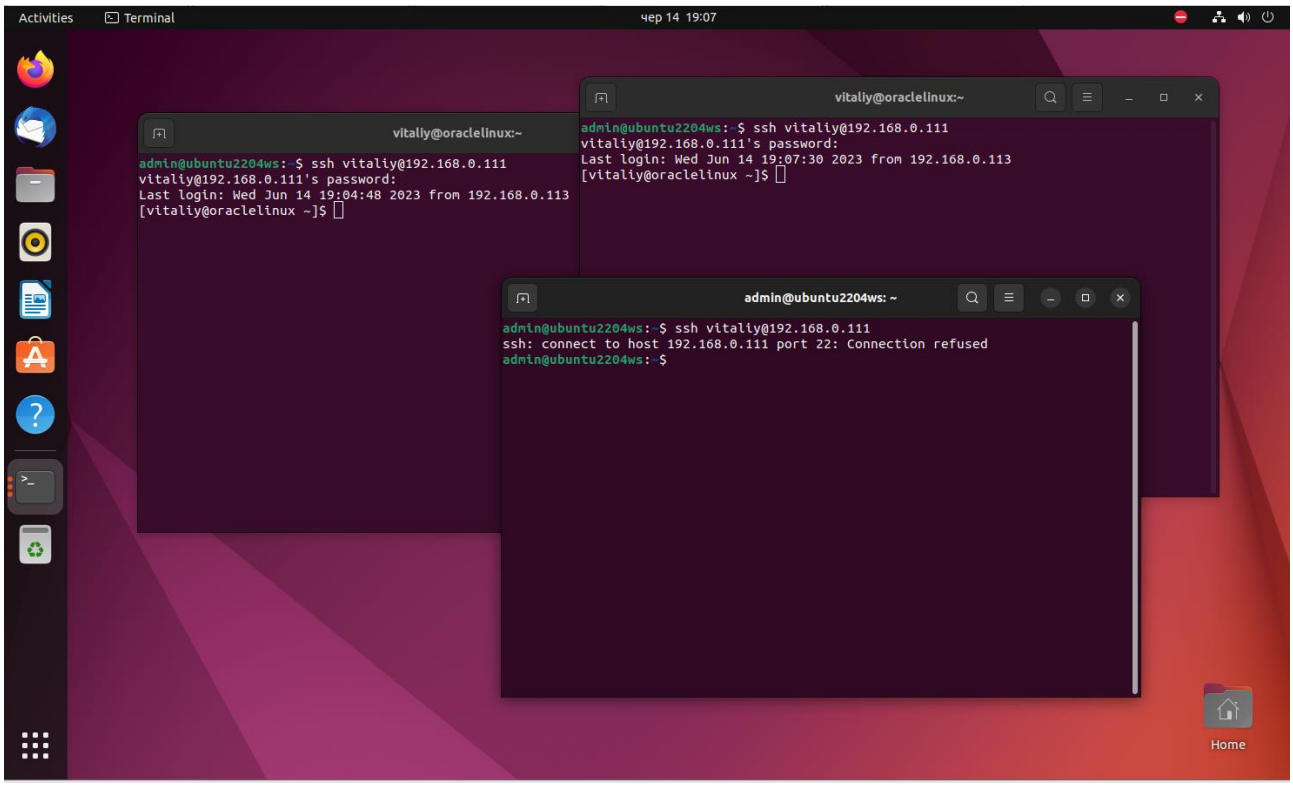

Рисунок 3.5 - Перевірка працездатності базових правил nftables

Також на Oracle Linux можна побачити що IP адреса заблокована згідно написаних правил зі зворотнім відліком часу (Рис.3.6).

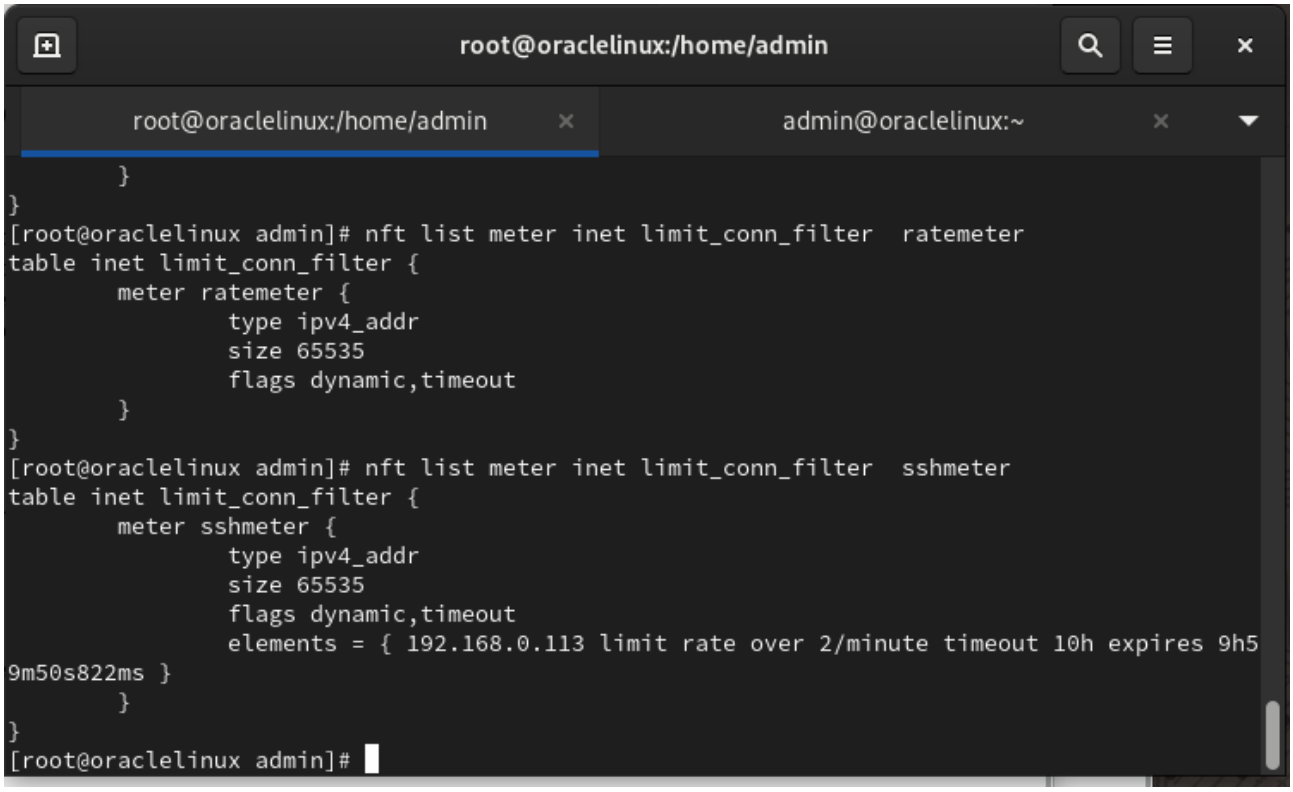

Рисунок 3.6 - Підтвердження блокування IP

Оптимальні обмеження будуть залежати від конкретних потреб і характеристик сервера. Важливо налаштувати обмеження таким чином, щоб воно забезпечувало достатній рівень захисту без негативного впливу на легітимний трафік і продуктивність системи.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при термічних опіках

Долікарська допомога при термічних опіках є важливим кроком у зменшенні наслідків та запобіганню подальших ускладнень. Термічні опіки відбуваються, коли шкіра або м’які тканини пошкоджуються високими температурами, які можуть виникнути в результаті вогню, гарячих речовин або пари, сонячного випромінювання або інших джерел тепла. Опіки можуть бути серйозними і потребують негайного медичного втручання.

При наданні домедичної допомоги враховувати ступінь пошкодження шкіри та м’яких тканин:

1) I ступінь (еритема) - почервоніння шкіри, набряклість і біль.

2) II ступінь (утворення пухирів) - сильний біль із інтенсивним почервонінням, відшаруванням епідермісу з утворенням міхурів, наповнених рідиною.

3) III ступінь: пошкодження всієї товщі шкіри з утворенням щільного струпу, під яким перебувають ушкоджені тканини.

4) IV ступінь (обвуглення): пошкодження всієї товщі шкіри з ушкодження м’язів, сухожиль, кісток.

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 термічних опіках. У цьому порядку термін «термічний опік» вживається у такому значенні - це невідкладний стан, спричинений дією високих температур, в результаті чого виникає пошкодження шкіри та м’яких тканин. [17].

Надання домедичної допомоги постраждалим при термічних опіках передбачає такі кроки:

1) Переконайтеся у відсутності небезпеки для себе, оточуючих та постраждалого.
2) Припиніть дію високої температури на постраждалого та, при необхідності, зніміть тліючий одяг.

3) Зніміть прикраси, які можуть бути на ділянці опіку.
4) Заспокойте постраждалого та поясніть йому свої подальші дії.
5) Зателефонуйте на екстрений номер, щоб здійснити виклик екстреної медичної допомоги, та слухайте вказівки диспетчера.
6) Охолодіть місце опіку протягом щонайменше 20 хвилин, промиваючи його водою кімнатної температури. Це важливо, якщо площа опіку не перевищує 20% у дорослих або 10% у дітей.
7) Після охолодження накладіть на місце опіку чисту, стерильну суху марлеву пов'язку. Пов'язка не повинна надмірно тиснути на м'які тканини.
8) У разі наявності міхурів не пошкоджуйте їх. Якщо випадково міхур пошкоджено, накладіть пов'язки, як описано вище.
9) Якщо опіки охоплюють більше ніж 20% площі тіла у дорослих або 10% у дітей, накрийте постраждалого термопокривалом або покривалом;
10) Забезпечте постійний нагляд за постраждалим до прибуття бригади швидкої медичної допомоги.
11) У разі погіршення стану постраждалого до прибуття бригади швидкої медичної допомоги повторно зателефонуйте диспетчеру.
12) Якщо можливо, зберіть якомога більше інформації про обставини отримання травми від постраждалого або свідків. Передайте цю інформацію фахівцям бригади швидкої медичної допомоги або диспетчеру.

Це загальна послідовність дій, яку слід виконати, але завжди важливо дотримуватись інструкцій медичних фахівців та адаптувати допомогу до конкретної ситуації. Виконання цих кроків допоможе забезпечити постраждалому першу необхідну допомогу та зберегти його життя до прибуття медичних фахівців.

4.2 Вимоги до профілактичних медичних оглядів для працівників ПК

Працівники, які використовують у своїй роботі персональні комп'ютери, підлягають обов'язковим медичним оглядам. Але водночас згідно з роз'ясненням МОЗ від 20.01.2006 р. № 05.0101-18-58/21 проведення обов'язкових медоглядів поширюється на роботу з комп'ютерами на основі електронно-променевих трубок та не поширюється на роботу з рідкокристалічними терміналами.

У сучасних умовах важко знайти галузь економіки, де б не використовувалися комп'ютери. Бюджетні організації не є винятком, адже в них за комп'ютерами працюють бухгалтери, секретарі, економісти, юристи, оператори комп'ютерного набору тощо.

Обов'язковість проведення медичних оглядів для працівників, які працюють за електронно-обчислювальними машинами, передбачена розд. 6 Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 р. № 7 (далі — ДСанПіН № 7).

Так, працюючі з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ) підлягають обов'язковим медичним оглядам: попереднім — при влаштуванні на роботу і періодичним — протягом трудової діяльності (п. 6.1 розд. 6 ДСанПіН № 7).

Такі медичні огляди проводяться відповідно до вимог Порядку проведення медичних оглядів працівників певних категорій, затвердженого наказом МОЗ від 21.05.2007 р. № 246 (далі — Порядок № 246).

Періодичні медичні огляди мають проводитися раз на два роки комісією в складі терапевта, невропатолога та офтальмолога.

До складу комісії, що проводить попередні та періодичні медичні огляди, при необхідності (за наявності медичних показань) можуть залучати лікарів інших спеціальностей.

Основними критеріями оцінки придатності до роботи з ВДТ ЕОМ і ПЕОМ мають бути показники стану органів зору: гострота зору, показники рефракції, акомодації, стану бінокулярного апарату ока тощо. При цьому також враховується стан організму в цілому.

У розд. 6 ДСанПіН № 7 передбачено й протипоказання для роботи за комп'ютерами. До них відносять усі хронічні форми психічних захворювань, ендокринні захворювання, тяжкий ступінь бронхіальної системи, гіпертонічна хвороба III стадії та інші захворювання.

До відома зазначимо, що монітор або дисплей — це електронний пристрій для відображення інформації. Комп'ютерні монітори бувають кількох типів:

- на основі електронно-променевої трубки(CRT);
- рідкокристалічні(LCD, TFT як підвид LCD);
- плазмові;
- проекційні;
- OLED-монітори.

Плазмові і проекційні монітори використовують там, де потрібен великий розмір екрану (діагональ метр і більше).

На питання чи необхідно проходити обов'язкові медогляди працівникам, які працюють за більш сучасними «тонкими» моніторами було надано роз'яснення заступника Головного державного санітарного лікаря України в листі від 20.01.2006 р. № 05.01.01-18-58/21. У цьому листі зазначено, що ДСанПіН № 7 поширюються на ВДТ усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок, тому вимога щодо проведення попередніх та періодичних медичних оглядів на працюючих з рідкокристалічними відеотерміналами ЕОМ машин у цьому випадку не діє.

Нагадаємо деякі правила проведення обов'язкових медоглядів.

Як уже було зазначено вище, працівники, які працюють за комп'ютерами, підлягають обов'язковим попередньому та періодичним медоглядам.

Попередній медичний огляд проводиться під час прийняття на роботу з метою:

- визначення стану здоров'я працівника і реєстрації вихідних об'єктивних показників здоров'я та можливості виконання без погіршення стану здоров'я про- фесійних обов'язків в умовах дії конкретних шкідливих та небезпечних факторів виробничого середовища і трудового процесу;

- виявлення професійних захворювань (отруєнь), що виникли раніше при роботі на попередніх виробництвах, та попередження виробничо зумовлених і професійних захворювань (отруєнь).

Періодичні медичні огляди проводяться з метою:

- своєчасного виявлення ранніх ознак гострих і хронічних професійних за- хворювань (отруєнь), загальних та виробничо зумовлених захворювань у працівників;

- забезпечення динамічного спостереження за станом здоров'я працівників в умовах дії шкідливих та небезпечних виробничих факторів і трудового процесу;

- вирішення питання щодо можливості працівника продовжувати роботу в умовах дії конкретних шкідливих та небезпечних виробничих факторів і трудо- вого процесу;

- розробки індивідуальних та групових лікувально-профілактичних та ре- абілітаційних заходів працівникам, що віднесені за результатами медичного огляду до групи ризику;

- проведення відповідних оздоровчих заходів.

Заклади державної санітарно-епідеміологічної служби щорічно за заявкою роботодавця (його представника), за участю представника первинної профспілкової організації або уповноваженої працівниками особи, визначають категорії працівників, які підлягають попередньому (періодичним) медичному огляду та до 1 грудня складають Акт визначення категорій працівників, які підлягають попередньому (періодичним) медичному огляду за формою, зазначеною у додатку 1 до Порядку № 246.

Потім на підставі зазначеного Акта роботодавець складає протягом місяця у чотирьох примірниках поіменні списки працівників, які підлягають

періодичним медичним оглядам. Форма таких списків наведена у додатку 2 до Порядку № 246.

Ці списки на паперовому та електронному носіях узгоджують у санітарно-епідеміологічній станції. Один примірник списку залишається на підприємстві (у відповідальної за організацію медогляду посадової особи), другий надсилається до закладу охорони здоров'я, третій — до закладу державної санітарно- епідеміологічної служби, четвертий — до робочого органу виконавчої дирекції Фонду соціального страхування від нещасних випадків на виробництві та професійних захворювань.

Отже, як видно, списки працівників, які мають пройти медогляди, надаються та узгоджуються з санепідслужбою.

Для проведення попереднього (періодичних) медичного огляду працівників роботодавець повинен укласти або вчасно поновити договір із закладом охорони здоров'я та надати йому список працівників, які підлягають попередньому (періодичним) медичному огляду.

Питання придатності до роботи в кожному окремому випадку вирішується індивідуально з урахуванням особливостей функціонального стану організму (характеру, ступеня прояву патологічного процесу, наявності хронічних захворювань), умов праці та результатів додаткових методів обстеження.

При цьому кожен лікар, який бере участь в обстеженні пацієнта, дає висновок щодо стану здоров'я працівника, підтверджує його особистим підписом та особистою печаткою, бере участь в остаточному обговоренні придатності обстежуваної особи до роботи в обраній професії та в разі необхідності визначає лікувально-оздоровчі заходи.

За результатами періодичних медичних оглядів (протягом місяця після їх закінчення) комісія з проведення медичних оглядів закладів охорони здоров'я оформляє Заключний акт за результатами періодичного медичного огляду працівників. Форма цього акта наведена у додатку 9 до Порядку № 246 (ср. 025069200). Такий акт складається у шести примірниках — один примірник залишається в закладі охорони здоров'я, що проводив медогляд, інші надаються

роботодавцю, представнику профспілкової організації або вповноваженій працівниками особі, профпатологу, закладу державної санітарно-епідеміологічної служби, робочому органу виконавчої дирекції Фонду соціального страхування від нещасних випадків на виробництві та професійних захворювань.

Також зазначимо, що роботодавець зберігає за працівником на період проходження медогляду місце роботи (посаду) і середній заробіток та за результатами медичного огляду інформує працівника про можливість (неможливість) продовжувати роботу за професією (п. 2.21 Порядку № 246).

Наприкінці також звернемо вашу увагу на те, що право на додаткову відпустку за роботу на комп’ютері мають усі працівники, незалежно від того, за якими моніторами вони працюють.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено дослідження методів та механізмів забезпечення безпеки операційної системи Linux, зокрема Oracle Linux. Були розглянуті основні загрози безпеки, такі як віруси, шкідливі програми, хакерські атаки та витоки інформації, а також досліджені методи їх протидії.

Для забезпечення безпеки в Linux були розглянуті різні механізми та методи, зокрема управління доступом, аутентифікація та авторизація, мережева безпека, захист даних та захист від шкідливих програм. Були розглянуті такі інструменти, як ClamAV та Rootkit Hunter для виявлення вірусів та шкідливого програмного забезпечення, а також SELinux та ACL для управління доступом до ресурсів системи.

Для моніторингу та аудиту подій були розглянуті інструменти, такі як syslog і AIDE. syslog дозволяє логувати події в операційній системі, що допомагає виявити небезпечну або підозрілу активність. AIDE дозволяє моніторити цілісність файлів у системі та перевіряти їх хеш-суми для виявлення змін або підозрілих модифікацій.

Також було розглянуто методи захисту сервера SSH, включаючи використання безпечних алгоритмів шифрування та хешування, аутентифікацію за допомогою ключів та налаштування фаєрвола nftables для обмеження кількості одночасних з'єднань та контролю мережевого трафіку.

Було розглянуто використання LUKS та EncFS, яке допомагає забезпечити високий рівень безпеки даних в операційній системі Linux. Шифрування дискових томів та файлових систем дозволяє захистити інформацію від несанкціонованого доступу, зберігаючи її конфіденційність. Ці механізми можуть бути використані як на рівні індивідуальних користувачів, так і на рівні підприємств та організацій, де безпека даних має велике значення.

Загалом, безпека операційної системи Linux є надзвичайно важливою задачею в сучасному інформаційному середовищі. Розуміння загроз безпеки та

використання ефективних методів та механізмів захисту є ключовими аспектами для забезпечення безпеки в Linux.

Важливо пам’ятати, що безпека є постійним процесом, і вона вимагає постійного оновлення та вдосконалення. Розуміння принципів безпеки та використання ефективних методів та механізмів є ключовими для забезпечення безпеки операційної системи Linux.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Linux [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.linux.com/training-tutorials/
2. Architecture of Linux [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.javatpoint.com/architecture-of-linux
3. Linux Threats [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.vskills.in/certification/tutorial/linux-threats/
4. Oracle Linux 9 [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.oracle.com/en/operating-systems/oracle-linux/9/
5. Using PAM [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – https://docs.oracle.com/cd/E19683-01/817-0365/pam-1/index.html
6. Linux Firewall [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://oracle-base.com/articles/linux/linux-firewall
7. The netfilter.org project [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://nftables.org/
8. Configuring SSL [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.oracle.com/cd/E16671_01/bh.200/e16641/ssl.htm
9. Горбенко І. Д. Гриненко Т. О. Захист інформації в інформаційно-телекомунікаційних системах: Навч. посібник. Ч.1. Криптографічний захист інформації - Харків: ХНУРЕ, 2004 - 368 с.
10. Linux Unified Key Setup [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://oracle-base.com/articles/linux/luks-encrypted-file-systems
11. EncFS [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://wiki.archlinux.org/title/EncFS

12. ClamAV Documentation [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.rockylinux.org/guides/web/apache_hardened_webserver/rkhunter/
13. Rootkit hunter [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.clamav.net/
14. Linux System Log Files [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://oracle-base.com/articles/linux/linux-system-log-files
15. Using Advanced Intrusion Detection Environment with Oracle Linux Automation Manager [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://www.oracle.com/a/ocom/docs/linux/using-advanced-intrusion-detection-environment.pdf
16. Connecting to Remote Systems With OpenSSH [Електронний ресурс] : [Інтернет-портал]. – Електронні дані. – Режим доступу: https://docs.oracle.com/en/operating-systems/oracle-linux/openssh/openssh-ConfiguringOpenSSHServer.html#config-openssh-server
17. 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