Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
« » червня 2021р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього рівня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Механізми захисту операційних систем з відкритим кодом»

Виконавець: студента IV курсу, групи КБ-41

_____ Кольцов Даніл Максимович
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Пархоменко І. І.	

Нормоконтроль	Даков С. Ю.	

Київ 2021

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
«10» жовтня 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-41** **Кольцову Данілу Максимовичу**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Механізми захисту операційних систем з відкритим кодом

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №2 від 08.10.2020 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Ядро Linux, модулі автентифікації, iptables, служби мережевої безпеки (NSPR NSS), SSL, репозиторій Linux.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з особливостями та вразливостями операційних систем з відкритим кодом. Реалізувати та формувати рекомендації до механізмів захисту Операційних систем з відкритим кодом.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Поєднання різних механізмів для захисту операційних систем з відкритим кодом та формування рекомендацій щодо їх використання.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання:12 жовтня 2020 року

Завдання видав _____ І. І. Пархоменко
(підпис) (ініціали, прізвище)

Завдання прийняла до виконання _____ Д. М. Кольцов
(підпис) (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	25.01.2021 – 27.01.2021	*виконано*
2	Аналіз літератури	28.01.2021 – 11.02.2021	*виконано*
3	Розгляд особливостей Linux	12.02.2021 – 24.02.2021	*виконано*
4	Дослідження основних вразливостей	25.02.2021 – 24.03.2021	*виконано*
5	Впровадження засобів та механізмів захисту доступності	25.03.2021 – 21.04.2021	*виконано*
6	Впровадження засобів та механізмів запобігання порушенню програмного забезпечення	22.04.2021 – 08.05.2021	*виконано*
7	Побудова репозиторія Linux з використанням механізмів захисту	09.05.2020 – 04.06.2021	*виконано*
8	Оформлення пояснювальної записки	05.06.2021 – 08.06.2021	*виконано*
9	Підготовка до захисту дипломної роботи	09.06.2021 – 21.06.2021	*виконано*

Завдання видав _____ І. І. Пархоменко
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ Д. М. Кольцов
(підпис) (ініціали, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2021 року

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 53 сторінок основного тексту. Список використаних джерел містить 78 найменування і займає 3 сторінки.

Метою даної роботи є реалізація механізму захисту операційних систем з відкритим кодом.

Об’єктом дослідження є процес захисту операційних систем з відкритим кодом.

Предметом дослідження є механізми захисту операційних систем з відкритим кодом.

У роботі проаналізована існуюча література, виконаний аналіз документів, порівняння, вивчення та узагальнення вітчизняної і зарубіжної практики з теми механізмів захисту операційних систем з відкритим кодом.

Розроблені рекомендації призначені для користувачів, що хочуть забезпечити безпеку своїх персональних даних в операційних системах з відкритим кодом.

Ключові слова: операційні системи з відкритим кодом, Unix-подібні операційні системи, збірка операційної системи на базі ядра Linux, конфігурація операційної системи, збірка ядра Linux, основні вразливості операційних систем, механізми захисту операційних систем, покращення входу, встановлення політик firewall, ssl, репозиторій оновлення Linux.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API	–	Application Programming Interface
chroot	–	change root
DNS	–	Domain Name Service
FQDN	–	Fully Qualified Domain Name
FTP	–	File Transfer Protocol
GCC	–	GNU Compiler Collection
GID	–	Group Identifier
IP	–	Internet Protocol
LSB	–	Linux Standard Base
NNTP	–	Network News Transport Protocol
PID	–	Process Identifier
UID	–	User Identifier
umask	–	user file-creation mask
USB	–	Universal Serial Bus
UTC	–	Coordinated Universal Time

ЗМІСТ

ВСТУП

Одна з головних цілей дипломної роботи - зрозуміти, як працює Linux зсередини та зробити його безпечним. Створення своєї системи допоможе дізнатися, як все працює разом і як кожен компонент системи взаємодіє з іншими.

Свій дистрибутив надає більш глибокий контроль. Не покладаючись на іншу реалізацію Linux. Можно диктувати кожен аспект своєї системи.

Свій дистрибутив дозволяє створити дуже компактну систему Linux. Коли встановлюється який-небудь дистрибутив примусово поставлять велику кількість програм, якими, швидше за все, ніколи не скористаєтеся. Вони будуть тільки витрачати системні ресурси.

Ще однією перевагою власної збірки Linux є безпека. У мережі є багато програмного забезпечення з відкритим кодом для поліпшення безпеки. При компіляції кожного компонента системи з вихідних кодів можена все перевірити і гнучка налаштувати. Тепер не треба чекати, коли хтось скомпілює пакет з необхідними виправленнями. Якщо не вивчите патч і не застосуєте його самостійно, немає гарантій, що новий пакет буде зібраний коректна і усуне проблему.

Є ще маса причин за якими слід виконувати збірку власної системи. В остаточному підсумку варто освітня мета є найважливішою.

РОЗДІЛ 1
ОСОБЛИВОСТІ ОПЕРАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ

Створення своєї операційної системи допоможе проаналізувати особливості, як все працює разом і як кожен компонент системи взаємодіє з іншими. Збірка системи Linux буде виконана за допомогою вже встановленого дистрибутива Linux.

1.1 Розділи диска і файлова система

Як більшість інших операційних систем, Linux зазвичай встановлюється на виділений розділ. Рекомендований підхід до створення системи Linux, полягає в використанні доступного порожнього розділу або, якщо є розділений простір, то можна використовувати його.

Мінімальна система вимагає розділу ємністю 6 гігабайт. Цього розміру буде досить для зберігання всіх архівів з вихідними кодами всіх пакетів і їх подальшої компіляції. Велика частина простору буде використана для тимчасового зберігання і для її подальшого розширення. При компіляції пакетів потрібно багато додаткового дискового простору, яке буде звільнено після установки пакета.

Оперативної пам'яті може бути не достатньо, рекомендується використовувати невеликий дисковий розділ swap. Він буде використовуватися ядром для зберігання рідка використовуваних даних і звільняти оперативну пам'ять для активних процесів. Розділ swap для системи Linux можна використовувати той, який використовується хост-системою, і створювати новий розділ - необов'язково.

Розділення диска робиться утилітами як cfdisk або fdisk, за допомогою опції командного рядка з іменем жорсткого диска, на якому буде створено новий розділ, наприклад /dev/sda для основного дисковода.

За замовчуванням для більшості дистрибутивів створення розділу полягає в використанні всього диска, за винятком одного невеликого розділу підкачки. Це не є оптимальним для Linux з кількох причин. Це зменшує гнучкість, робить спільне використання даних в різних розділах або збірках Linux складніше, ускладнює

процес резервного копіювання, і вимагає більше часу і може витрачати дисковий простір менш ефективно.

Кореневий розділ Linux розміром в 10 гігабайт є хорошим компромісом для багатьох систем. Такого розміру буде досить для збірки Linux і більшості пакетів Linux, але при цьому, він досить малий.

Багато дистрибутиви автоматично створюють розділ підкачки. Як правило, рекомендований розмір - в два рази більше, ніж обсяг оперативної пам'яті.

Існує кілька інших розділів, які не потрібні, але повинні враховуватися при проектуванні і розбивці розділів:

/Boot - Настійно рекомендується. Використовується цей розділ для зберігання Linux ядер і завантажувального інформації.

/Home - Настійно рекомендується. Розділ для домашнього каталогу користувачів для декількох дистрибутивів або збірок Linux. Розмір як правило необхідно вказати досить великий, в залежності від доступного місця на диску.

/Usr - Окремий розділ /usr зазвичай використовується для тонких клієнтів і бездискових робочих станцій. Використання цього каталогу як окремого розділу - зазвичай не потрібно для Linux. Розмір близько п'яти гігабайт підійде для більшості установок.

/Opt - Цей каталог буде корисний в основному для Linux, де множинна установка великих пакетів, таких як Gnome або KDE може бути виконана не в ієрархію каталогів /usr. Розміру від п'яти до десяти гігабайт буде досить.

/Tmp - Окремий каталог /tmp зустрічається рідко, і корисний для роботи тонких клієнтів. Якщо використовується цей розділ, він може не перевищувати пари гігабайт.

/Usr/src - Цей розділ корисно мати для збереження вихідних файлів BLinux і виконувати обмін між збірками Linux. Він також може використовуватися як місце для складання пакетів BLinux. 30 - 50 гігабайт буде досить.

Linux може використовувати будь-які файлові системи, які можуть бути розпізнані ядром Linux. В основному це файлові системи ext3 і ext4. Вибір файлової

системи може залежати від характеристик збережених файлів і розміру розділу. наприклад:

ext2 підходить для невеликих розділів, які оновлюються нечасто, наприклад для розділу / boot.

ext3 Це оновлена файлова система ext2, що включає в себе журнал, для відновлення в разі некоректного вимкнення. Зазвичай використовується як файлова система загального призначення.

ext4 Є на даний момент останньою версією файлової системи сімейств ext. Вона підтримує багато нових можливостей таких як nano-second мітки часу, створення і використання дуже великих файлів (16 Терабайт), і поліпшення швидкодії.

Linux припускає, що коренева файлова система буде ext4. Щоб виконати форматування розділу використовуючи цю файлову систему, треба виконати наступну команду (рис 1.1):

```
mkfs -v -t ext4 /dev/<xxx>
```

Рисунок 1.1 – Команда форматування розділу

У багатьох пунктах присутня змінна оточення Linux. До виконання процесу складання, необхідно переконатися в тому, що змінна визначена. Змінна повинна зберігати шлях до каталогу, де буде виконуватися процес складання Linux системи. Якщо збірка виконується на окремому розділі, каталог може бути точкою монтування розділу. Визначте каталог, де буде зберігатися система Linux і виконайте наступну команду (рис 1.2):

```
export Linux=/mnt/Linux
```

Рисунок 1.2 – Команда экспорту змінної

Тепер, необхідно зробити розділи доступними. Щоб це зробити, необхідно примонтувати розділи в обрані точки монтування.

Для створення точки монтування файлової системи Linux, виконавши команду (рис 1.3):

```
mkdir -pv $Linux
mount -v -t ext4 /dev/<xxx> $Linux
```

Рисунок 1.3 – Команда створення папки та монтування

Де <xxx> найменуванням форматированого розділу Linux.

1.2 Пакети і патчі для системи

Для того щоб завантажені пакети і патчі десь зберегти потрібен робочий каталог, в якому можна буде розпаковувати пакети і виконувати їх налаштування і компіляцію. Каталог $Linux/sources може бути використаний як місце для зберігання, а також як місце для настройки і компіляції. Використовуючи цей каталог, необхідні елементи будуть розташовані і доступні на всіх етапах створення системи Linux.

Щоб створити такий каталог, треба виконати наступну команду як користувач root, до початку процесу завантаження пакетів і патчів (рис1.4):

```
mkdir -v $Linux/sources
```

Рисунок 1.4 – Команда створення папки

Треба зробити цей каталог доступним для запису і липким. «Липким» означає, що навіть якщо у деяких користувачів є дозвіл на запис в цей каталог, то тільки власник зможе видалити вміст каталогу. Наступна команда дозволить це зробити (рис 1.5):

```
chmod -v a+wt $Linux/sources
```

Рисунок 1.5 – Команда зміни прав доступу

Найпростіший спосіб завантаження всіх необхідних пакетів і патчів - скористатися файлом wget-list . Далі його можна передати як параметр програмі wget, наприклад (рис 1.6):

```
wget --input-file=wget-list --continue --directory-prefix=$Linux/sources
```

Рисунок 1.6 – Команда завантаження файлів

Завантажу пакети [1] із зазначеними версіями. До пакетів будуть потрібні патчі. Вони виправляють помилки в пакетах які повинні бути виправлені власником пакета. Патчі також вносять невеликі модифікації щоб полегшити роботу з пакетами. Наступні патчі [2] необхідні для складання системи Linux.

1.3Налаштування робочого оточення

Треба створити каталог в $Linux для установки тимчасового набору інструментів, і додати непривілейованого користувача, щоб знизити ризик, і налаштувати оточення для цього користувача.

Всі програми які будуть скомпільовані будуть встановлені в каталог $Linux/tools щоб можна було залишити їх окремо від збірки кінцевої системи. Програми які будуть скомпільовані - тимчасові інструменти і не будуть входити в кінцеву збірку Linux системи. Після використання тимчасових інструментів, від них можна позбутися. Використання каталогу $Linux/tools для зберігання тимчасового набору інструментів також корисно для того, щоб не засмічувати робочі каталоги хост-системи.

Треба створити каталог, виконавши команду від користувача root:

```
mkdir -v $Linux/tools/
```

Рисунок 1.7 – Команда створення папки

Наступним кроком, буде створення символічного посилання /tools в хост-системі, яка буде вказувати на створений каталог з тимчасовими інструментами. Запустивши команду від користувача root (рис 1.8):

```
ln -sv $Linux/tools/
```

Рисунок 1.8 – Команда створення посилання

Створена символічна посилання дозволяє виконати компіляцію тимчасового набору інструментів за посиланням /tools.

Коли користувач знаходиться в системі під користувачем root, одна єдина помилка може призвести до пошкодження або поломки хост-системи. Можна використовувати довільного користувача, але для спрощення налаштування чистого робочого оточення створіть нового користувача з ім'ям Linux як члена групи Linux і використовуйте цього користувача на час всього процесу установки тимчасового набору інструментів. Виконайте команду від користувача root (рис1.9):

```
groupadd Linux
useradd -s /bin/bash -g Linux -m -k /dev/null Linux
```

Рисунок 1.9 – Команда створення группи та користувача

Значення параметрів командного рядка:

-s /bin/bash - встановлює bash оболонкою за замовчуванням для користувача Linux;

-g Linux - опція додає користувача Linux в створену групу Linux;

-m - створює домашній каталог для користувача Linux;

-k /dev/null- запобігає можливому копіювання файлів з заздалегідь визначеного набору каталогів (за замовчуванням /etc/skel), змінивши місце розташування введення на спеціальне null пристрій;

Linux - Це найменування створеного користувача і групи.

Щоб увійти в систему як користувач Linux (на відміну від зміни користувача Linux коли користувач авторизован як користувач root, для якого не потрібно, щоб користувач Linux мав пароль), треба створити для облікового запису пароль (рис 1.10):

```
passwd Linux
```

Рисунок 1.10 – Команда створення группи та користувача

Наддам Linux повний доступ до $Linux/tools зробивши користувача Linux власником каталогу (рис1.11):

```
chown -v Linux $Linux/tools
```

Рисунок 1.11 – Команда зміни прав доступу

Якщо був створений окремий робочий каталог, як пропонується, треба надати користувачеві Linux права на наступний каталог (рис 1.12):

```
chown -v Linux $Linux/sources
```

Рисунок 1.12 – Команда зміни прав доступу

Далі, треба виконати вхід як користувач Linux. Цю дію можна виконати в графічній оболонці, використовуючи віртуальний термінал, або в звичайній користувальницької середовищі (рис 1.13):

```
su - Linux
```

Рисунок 1.13 – Команда зміни прав доступу

Аргумент «-» зраджує значення для команди su для початку запуску login-оболонки, на відміну від звичайної.

Налаштуємо оточення. Створимо два файли для оболонки bash. При вході в систему як користувач Linux виконайте наступну команду для створення нового файлу .bash_profile (рис 1.14):

```
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
```

Рисунок 1.14 – Команда для створення нового файлу .bash_profile

Коли виконано вхід під користувачем Linux, при ініціалізації оболонки login, буде читати дані з файлу /etc/profile хост-системи (який можливо буде містити деякі настройки і додаткові змінні), і потім дані з файлу .bash_profile. Команда exec env -i .../bin/bash в файлі .bash_profile замінює запущену оболонку нової повністю порожній середовищем, крім таких змінних, як HOME, TERM, і PS1. Такий підхід гарантує, що небажані і потенційно небезпечні змінні оточення хост-системи не потраплять в середу збірки.

Новий екземпляр оболонки - це non-login оболонка, яка не виконує читання файлу /etc/profile або .bash_profile, але виконує читання з файлу .bashrc. Створимо файл .bashrc (рис 1.15):

```
set +h
umask 022
Linux=/mnt/Linux
LC_ALL=POSIX
Linux_TGT=$(uname -m)-Linux-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export Linux LC_ALL Linux_TGT PATH
```

Рисунок 1.15 – Команди налаштування робочого оточення

Команда set + h відключає хеш-функцію bash. Хешування корисно коли bash використовує хеш-таблицю для зберігання повного шляху виконуваних файлів щоб уникнути пошуку в змінної оточення PATH при кожному зверненні. Однак нові

інструменти повинні бути використані відразу після їх установки. При виключенні хеш-функції оболонка буде завжди шукати змінну оточення PATH коли програму необхідно запустити. Таким чином, оболонка буде виконувати пошук скомпільованих інструментів в каталозі $Linux/tools, як тільки вони стануть доступні без запам'ятовування попередньої версії тієї ж програми в іншому розташуванні.

Вказівка для значення 022 маски створення файлів користувача (umask) для створення нових файлів і каталогів дозволяє виконувати запис тільки їх власнику, але вони залишаться доступними на читання і виконання для інших користувачів (за умови, що режими за замовчуванням використовують системний виклик open(2), нові файли отримають дозвіл 644 а каталоги 755).

Змінна оточення Linux повинні вказувати на обрану точку монтування.

Змінна LC_ALL управляє локалізацією деяких програм, і формує повідомлення відповідно до угод локалізацій зазначеної країни. Якщо вказати значення LC_ALL в «POSIX» або «C» гарантує, що все буде працює так, як очікується в chroot оточенні.

Змінна Linux_TGT містить сумісне опис комп'ютера, яка буде використовуватися при складанні крос-компілятора і компоновальника і потім при крос-компіляції тимчасового набору інструментів.

Вказівка значення /tools/bin перед стандартним вмістом змінної PATH дає можливість оболонці відразу використовувати програми і бібліотеки, які були встановлені в цей каталог. У поєднанні з відключенням хеширования, знижується ризик, що старі програми будуть використані хост-системою коли ті ж програми будуть доступні в середовищі оточення.

Щоб повністю підготувати середовище для тимчасових інструментів, треба поставити джерело щойно створеного профілю користувача (рис 1.16):

```
source ~/.bash_profile
```

Рисунок 1.16 – Команда щоб поставити дрежерело профілю

1.4 Збірка тимчасової системи

Опис процесу створення мінімальної Linux системи. Ця система буде містити тільки ті інструменти, які будуть потрібні для збірки кінцевої системи Linux і дозволить працювати з зручними настройками, ніж звичайна мінімальна середу.

Необхідно виконати два кроки для збірки мінімальної системи. Першим кроком буде виконано складання незалежного від хост системи набору інструментів (компілятор, асемблер, компонувальник, бібліотеки, і ще кілька корисних утиліт). Другим кроком, з використанням цих інструментів будуть скомпільовані інші необхідні пакети.

Всі пакети будуть встановлені в каталог $Linux/tools щоб зберігати їх окремо від хост системи і кінцевої системи Linux.

Головним завданням є створення тимчасової області яка містить необхідний набір інструментів, який може бути ізольований від хост-системи. Використання команди chroot в останніх главах забезпечить чисте оточення і безпроблемну збірку кінцевої системи Linux.

Ключові технічні вказівки методу збирання:

Тимчасові бібліотеки будуть скомпільовані крос-компілятором, тому що він не залежить від хост-системи. Такий метод виключає потенційне забруднення цільової системи, зменшуючи ймовірність того, що будуть включені будь-які заголовки та бібліотеки хост-системи в тимчасовий набір інструментів. Крос компіляція також дозволяє виконувати збірку як 32-бітових, так і 64-бітових бібліотек на 64-бітному сумісному апаратному забезпеченні.

Акуратна маніпуляція з вихідними кодами пакету GCC буде вказувати компілятору якої цільової динамічний завантажувач необхідно використовувати.

Пакет Binutils встановлюється першим, тому що команда configure запускає GCC і Glibc і виконує різні функціональні тести асемблера і компонувальника для визначення які функції програмного забезпечення включити або вимкнути. Некоректно налаштований GCC або Glibc можуть привести до непрацездатності

всього тимчасового набору інструментів, і така проблема може проявитися тільки в кінці збірки дистрибутива.

Пакет Binutils виконує установку нового асемблера і компонувальника в два каталоги: /tools/bini /tools/$Linux_TGT/bin. Інструменти в одному каталозі жорстко пов'язані. Важливою особливістю компоновщика є порядок пошуку бібліотек. Детальну інформацію можна отримати за допомогою команди ld з аргументом ld --verbose | grep SEARCH. Наприклад команда ld --verbose | grep SEARCH покаже поточні шляху пошуку і їх порядок.

Наступний пакет який буде встановлений - GCC. Щоб дізнатися, який стандартний компонувальник **gcc** буде використовувати, запустіть команду: gcc -print-prog-name=ld.

Далі будуть встановлені ізольовані заголовки Linux API (Linux API headers). Це дозволить бібліотеці C (Glibc) взаємодіяти з функціями, які надає ядро Linux.

Наступний пакет який буде встановлений - Glibc. Необхідні компоненти для збірки Glibc - це компілятор, інструменти для роботи з бінарними файлами, та заголовки ядра Linux (Linux API headers).

Під час другого проходу збірки пакета Binutils можно використовувати аргумент --with-lib-path щоб настроїти шляху пошуку для бібліотеки ld .

Для другого проходу збірки GCC, знадобиться модифікація вихідних кодів цього пакета щоб повідомити GCC використовувати новий, створений раніше динамічний компонувальник. Недотримання цієї вимоги призведе до того, що програми будуть використовувати динамічний компонувальник хост системи, який знаходиться в каталозі /lib, що суперечить завдання ізоляції тимчасового набору інструментів. Починаючи з цього моменту ядро тимчасового набору інструментів повністю самодостатня і незалежна.

При вході в середу chroot перший основний пакет, який буде скомпільовано - це пакет Glibc, в силу згаданих вище аспектів самодостатності. Як тільки Glibc буде встановлений в каталог /usr треба виконати перемикання з налаштувань за замовчуванням тимчасового набору інструментів.

Отже скомпілюю та встановлю ці пакети: Binutils-2.32 - Прохід 1, GCC-9.2.0 - Прохід 1, Заголовки Linux-5.2.8, Glibc-2.30, Libstdc ++ з пакету GCC-9.2.0, Binutils-2.32 - Прохід 2, GCC-9.2.0 - Прохід 2, Tcl-8.6.9, Expect-5.45.4, DejaGNU-1.6.2, M4-1.4.18, Ncurses-6.1, Bash-5.0, Bison-3.4.1, Bzip2-1.0.8, Coreutils-8.31, Diffutils-3.7, File-5.37, Findutils-4.6.0, Gawk-5.0.1, Gettext-0.20.1, Grep-3.3, Gzip-1.10, Make-4.2.1, Patch-2.7.6, Perl-5.30.0, Python-3.7.4, Sed-4.7, Tar-1.32, Texinfo-6.6, Xz-5.2.4.

Зараз, права на каталог $Linux/tools належать користувачеві Linux, який існує тільки в хост-системі. Якщо права на каталог залишити як є, файли будуть мати приналежність до призначеного для користувача ідентифікатором але без відповідної облікового запису. Це небезпечно, тому що в подальшому коли будуть створюватися нові облікові записи, ідентифікатор може бути кому небудь призначений, і права на цей каталог будуть йому надані, отже він зможе маніпулювати всім вмістом каталогу без будь-яких обмежень.

Щоб уникнути цієї проблеми, пізніше, можна додати користувача Linux в нову систему, коли буде створений файл /etc/passwd, подбайте призначити йому теж ім'я та ідентифікатор групи як в хост-системі. Найкраще змінити права на каталог $Linux/tools, та призначити їх користувачеві root, запустивши команду:

```
chown -R root:root $Linux/tools
```

Незважаючи на те, що каталог $Linux/tools може бути видалений, як тільки збірка системи Linux буде завершена, його можна зберегти для збірки іншої системи Linux.

1.5 Установка основного системного програмного забезпечення

Описання процесу складання кінцевої системи Linux за допомогою раніше встановленого тимчасового набору інструментів. За допомогою команди chroot буде виконаний вхід в тимчасову міні Linux систему (тимчасовий набір інструментів), далі будуть виконані остаточні підготовчі роботи, а потім будуть виконані установки необхідних пакетів.

Порядок установки пакетів дуже важливий. Необхідно суворо дотримуватися послідовність збірки і установки пакетів, для гарантії того, що встановлюється пакет не буде мати посилання на каталог /tools.

Треба виконати команду, щоб створити структуру каталогів, куди буде виконано монтування віртуальних файлових систем (рис 1.17):

```
mkdir -pv $Linux/{dev,proc,sys,run}
```

Рисунок 1.17 – Команда для створення каталогів

Коли ядро виконує завантаження системи, йому потрібна наявність декількох пристроїв, таких як console і null. Ці вузли повинні бути створені на диску, для того, щоб вони були доступні після того, як запуститься udevd і додатково, коли Linux запуститься з init=/bin/bash. Для того, щоб створити такі пристрої, виконаю команду (рис 1.18):

```
mknod -m 600 $Linux/dev/console c 5 1
mknod -m 666 $Linux/dev/null c 1 3
```

Рисунок 1.18 – Команда для створення пристроїв

Рекомендований метод заповнення каталогу (посилання на пристрої) /dev – примонтувати віртуальну файлову систему (таку, як tmpfs) в каталог /dev, і дозволити пристроям динамічно створюватися на цих файлових системах по мірі їх виявлення або можливості доступу до них. Створення пристроїв зазвичай виконується під час процесу завантаження програми Udev. Але оскільки в новій системі ще відсутня ця програма, отже пристрої додані не будуть. Необхідно вручну зв'язати пристрої з каталогу /dev хост-системи. Прив'язка - це особливий вид монтування, який дозволяє створювати дзеркало каталогу в зазначеній точці монтування в іншому місці розташування. Для того, щоб це зробити, скористаюсь командою (рис 1.18):

```
mount -v --bind /dev $Linux/dev
```

Рисунок 1.18 – Команда для монтування

Тепер, необхідно примонтувати віртуальні файлові системи (рис 1.19):

```
mount -vt devpts devpts $Linux/dev/pts -o gid=5,mode=620
mount -vt proc proc $Linux/proc
mount -vt sysfs sysfs $Linux/sys
mount -vt tmpfs tmpfs $Linux/run
```

Рисунок 1.19 – Команди для монтування

У деяких хост-системах, каталог /dev/shm є символічною посиланням на /run/shm. / run tmpfs був встановлений вище, тому потрібно створити тільки каталог (рис 1.19).

```
if [ -h $Linux/dev/shm ]; then
  mkdir -pv $Linux/$(readlink $Linux/dev/shm)
fi
```

Рисунок 1.19 – Скрипт для створення папок

Тепер, необхідно увійти в оточення Chroot, для початку побудови та встановлення остаточної системи Linux. Як користувач root виконаю наступну команду, для входу, з використанням створеного раніше набору тимчасових інструментів, які перебувають у тимчасовому каталозі tools (рис 1.20):

```
chroot "$Linux" /tools/bin/env -i \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='(Linux chroot) \u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
    /tools/bin/bash --login +h
```

Рисунок 1.20 – Скрипт для входу в оточення Chroot

Тепер немає необхідності використовувати змінну оточення Linux, тому що вся робота буде обмежена файлової системою Linux. Так відбувається, тому що оболонці Bash раніше повідомлявся в значенні змінної $Linux путь до кореня файлової системи, але на даний момент, вона є кореневим каталогом (/).

Це означає, що тимчасовий інструмент не буде більше використовуватися після установки його остаточної версії. Це відбувається, коли оболонка не запам'ятовує шлях до виконуваних бінарним файлам - з цієї причини, хешування вимкнено, зазначенням параметра +h до програми bash .

Коли оболонка запуститься, bash видає запрошення командного рядка виду I have no name!. Це нормально, тому що файл /etc/passwd ще не створений.

Саме час створити структуру каталогів для нової системи Linux. Необхідно виконати наступні команди, щоб створити стандартне дерево каталогів (рис 1.21):

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib/firmware,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -v  /usr/libexec
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -v  /usr/lib/pkgconfig
case $(uname -m) in
 x86_64) mkdir -v /lib64 ;;
esac
mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{color,misc,locate},local}
```

Рисунок 1.21 – Команди для створення стандартного дерево каталогів

Деякі програми використовують жорстко зашиті шляху до інших програм, які ще не встановлені. Щоб скорегувати цей недолік, потрібно створити ряд символічних посилань, які будуть замінені реальними файлами в процесі установки програм (рис 1.22).

```
ln -sv /tools/bin/{bash,cat,chmod,dd,echo,ln,mkdir,pwd,rm,stty,touch} /bin
ln -sv /tools/bin/{env,install,perl,printf}         /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1}                  /usr/lib
ln -sv /tools/lib/libstdc++.{a,so{,.6}}             /usr/lib
ln -sv bash /bin/sh
```

Рисунок 1.21 – Команди для створення посилань

Для здійснення можливості авторизації користувачем root від імені «root», повинні бути відповідні записи в файлах /etc/passwdi /etc/group.

Необхідно створити файл /etc/passwd (рис 1.22):

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/var/run/dbus:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
```

Рисунок 1.22 – Команди для створення файлів /etc/passwd

Фактичний пароль для root (символ «x» тут використовується тільки для заповнення) буде вказано пізніше.

Необхідно створити файл /etc/group (рис 1.22):

```
root:x:0:\nbin:x:1:daemon\nsys:x:2:\nkmem:x:3:\ntape:x:4:\ntty:x:5:\ndaemon:x:6:\
nfloppy:x:7:\ndisk:x:8:\nlp:x:9:\ndialout:x:10:\naudio:x:11:\nvideo:x:12:\nutmp:x:13:\nus
b:x:14:\ncdrom:x:15:\nadm:x:16:\nmessagebus:x:18:\ninput:x:24:\nmail:x:34:\nkvm:x:61:\
nwheel:x:97:\nnogroup:x:99: \nusers:x:999:\n
```

Рисунок 1.22 – Змісту файлу /etc/group

Створені групи не є частиною будь-якого стандарту. Ці групи дозволяють налаштувати Udev, і частково загальної конвенцією використовуваної поруч існуючих дистрибутивів Linux. Специфікація LSB (The Linux Standard Base, доступна по посиланню https://wiki.linuxfoundation.org/lsb/start) рекомендує, щоб крім групи root з ідентифікатором групи (GID), рівним 0, була присутня група bin з GID, рівним 1. Всі інші імена груп і ідентифікатори GID можуть вільно вибиратися системним адміністратором, оскільки добре написані програми не залежать від номерів GID, а використовують тільки ім'я групи.

Для того, щоб прибрати запрошення командного рядка «I have no name!», Необхідно запустити нову командну оболонку. Коли був встановлений пакет і файл /etc/group був створений, ім'я користувача і групи, тепер будуть працювати(рис 1.23):

```
exec /tools/bin/bash --login +h
```

Рисунок 1.23 – Команда для запуску командної оболонки

Зверніть увагу на використання аргументу +h. Він повідомляє програмі bash не використовувати власний механізм хешування шляхів. Без цього аргументу, bash буде запам'ятовувати шляху до двійковим файлів які виконувалися. Для того, щоб використовувати нові скомпільовані пакети, в міру їх установки, аргумент +h буде використаний протягом всієї глави.

Скомпілюю та встановлю ці пакети: API Linux, Man-pages, Glibc, Zlib, File, Readline, M4, Bc, Binutils, GMP, MPFR, MPC, Shadow, GCC, Bzip2, Pkg-config, Ncurses, Attr, Acl, Libcap, Sed, Psmisc, Iana-Etc, Bison, Flex, Grep, Bash, Libtool, GDBM, Gperf, Expat, Inetutils, Perl, XML::Parser, Intltool, Autoconf, Automake, Xz, Kmod, Gettext, Libelf, Libffi, OpenSSL, Python, Ninja, Meson, Coreutils, Check, Diffutils, Gawk, Findutils, Groff, GRUB, Less, Gzip, IPRoute2, Kbd, Libpipeline, Make, Patch, Man-DB, Tar, Texinfo, Vim, Procps-ng, Util-linux, E2fsprogs, Sysklogd, Sysvinit, Eudev.

1.6 Конфігурація системи

Завантаження Linux системи включає в себе кілька завдань. Процес спершу повинен примонтувати як віртуальні, так і реальні файлові системи, створити мережі, запустити служби, необхідні системою і виконати будь-які інші завдання, необхідні користувачу. Цей процес повинен бути організований для виконання завдань в правильному порядку і, в той же час, виконуватися як можна швидше.

SystemV - представляє класичний процес завантаження, який використовувався в Unix і Unix-подібних системах, таких як Linux з 1983 року. Він

складається з невеликої програми init, яка встановлює базові програми, такі як login (через getty) і запускає інші сценарії. Цей сценарій, як правило, називається rc, і управляє виконанням додаткових сценаріїв, необхідних для ініціалізації системи.

Програма init управляється файлом /etc/inittabі структурована на рівні запуску, які можуть виконуватися користувачем:

0 - Halt – зупинка

1 - Single user mode - Однокористувацький режим

2 - Multiuser, without networking - багато користувачів режим, без мережі

3 - Full multiuser mode - Повний багатокористувацький режим

4 - User definable - Режим, який визначається користувачем

5 - Full multiuser mode with display manager - Повний багатокористувацький режим з екранним менеджером

6 - reboot – перезавантаження

Недоліки:

Повільне завантаження. Середня швидкість завантаження Linux системи складає 8-12 секунд, якщо виміряти час з початку першого відображення запрошення командного рядка ядра. З'єднання з мережею як правило встановлюється приблизно 2 секунди після запрошення командного рядка.

Послідовна обробка завдань завантаження. Затримка в роботі будь-якого процесу, наприклад перевірка файлової системи, затримає відповідно і весь процес завантаження.

Не підтримує безпосередньо додаткові функції, такі як контроль груп (cgroups) і розкладу для кожного користувача.

Для додавання сценаріїв потрібні ручні, статичні послідовні рішення.

Треба встановити Linux-Bootscripts-20190524

Конфігурую мережу, у новій схемі іменування звичайні імена мережевих пристроїв будуть чимось на зразок enp5s0 або wlp3s0. Якщо ця угода про іменування не потрібно, традиційна схема або своя власна, може бути використана.

Традиційна схема іменування - eth0, eth1, і так далі, можуть бути встановлені шляхом додавання net.ifnames=0 в командний рядок ядра. Це підійде для систем, які

мають тільки мережеве пристрій одного типу. У ноутбуках часто є кілька мережевих з'єднань з іменами eth0 і wlan0. Ця схема цілком застосовна до них. Командний рядок передається в файлі конфігурації GRUB.

Які інтерфейси викликаються і вимикаються мережевим сценарієм, зазвичай залежить від файлів, які знаходяться в каталозі /etc/sysconfig/. Цей каталог повинен містити файл для кожного настроюваного інтерфейсу, наприклад ifconfig.xyz, де "xyz" повинен описувати мережеву карту. Ім'я інтерфейсу (наприклад, eth0) зазвичай є підходящим. Усередині цього файлу знаходяться атрибути цього інтерфейсу, такі як IP-адреси, маски підмережі і так далі. Необхідно, щоб назва файлу починалося з ifconfig .

Наступна команда створює приклад конфігураційного файлу для пристрою eth0 зі статичним IP-адресою рис(1.24):

```
cd /etc/sysconfig/
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
```

Рисунок 1.24 – Приклад конфігураційного файлу для пристрою eth0

Системі потрібні деякі засоби для отримання імен служби доменних імен (DNS) для перетворення доменних імен мережі Інтернет в IP-адреси і навпаки. Це досягається шляхом розміщення IP-адреси DNS-сервера, доступного від провайдера або адміністратора мережі, в /etc/resolv.conf. Створіть файл, виконавши такі дії (рис 1.25):

```
domain <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>
```

Рисунок 1.25 – Приклад конфігураційного файлу для пристрою/etc/resolv.conf

У процесі завантаження файл /etc/hostname використовується для установки імені хоста системи.

Треба створити файл /etc/hostname і напишіть в ім'я хоста, виконавши команду (рис 1.26):

```
echo "<Linux>" > /etc/hostname
```

Рисунок 1.26 – Команда для запису в файл

де <Linux> ім'я комп'ютера.

Треба створити файл /etc/hosts (рис 1.27):

```
127.0.0.1 localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters
```

Рисунок 1.27 – Зміст /etc/hosts

Значення <192.168.1.1>, <FQDN>, і <HOSTNAME> повинні бути змінені відповідно до власних уподобань та параметрів мережі (якщо є IP-адреса призначені системним / мережевим адміністратором і машина підключена до існуючої мережі.) Необов'язкові псевдоніми можуть бути опущені.

Програма оболонки /bin/bash використовує колекцію стартових файлів, щоб допомогти створити середовище для запуску. Кожен файл має певне використання і

може по-різному впливати на логін та інтерактивне середовище. Файли в /etc каталозі надають загальні налаштування. Якщо еквівалентний файл існує в домашньому каталозі, він може замінити загальні налаштування.

Інтерактивна оболонка для входу запускається після успішного входу, використовуючи /bin/login, зчитуючи /etc/passwd файл. Інтерактивна оболонка, що не входить в систему, запускається в командному рядку.

Файли /etc/profilei ~/.bash_profile зчитаються, коли оболонка викликається як інтерактивна оболонка.

Список усіх мов, підтримуваних Glibc, можна отримати, запустивши наступну команду (рис 1.28):

```
locale -a
```

Рисунок 1.28 – Команда для перегляду мов які підтримуються

Файл shells містить список оболонок входу в систему. Додатки використовують цей файл для визначення коректності оболонки. Для кожної оболонки повинна бути присутнім один рядок, що складається з шляху до файлу оболонки щодо кореня структури каталогів (/).

Наприклад, chsh звертається до цього файлу, щоб визначити, чи може непривілейований користувач змінити оболонку входу для свого облікового запису. Якщо ім'я команди не вказано, користувачеві буде відмовлено в зміні.

Ця вимога для таких додатків, як GDM, які не заповнюють браузер face, якщо він не вдається знайти /etc/shells, або FTP-демони, які традиційно забороняють доступ користувачам з оболонками, які не включені в цей файл.

/bin/sh

/bin/bash

1.7 Збірка ядра і установка завантажувача

Тут обговорюється створення файлу fstab, збірка ядра для нової системи Linux і установка завантажувача GRUB таким чином, щоб при початковому завантаженні, нову Linux систему можна було вибрати.

/etc/fstab файл використовується деякими програмами для визначення, де файлові системи повинні бути встановлені за замовчуванням, в якому порядку, і які повинні бути перевірені (на наявність помилок цілісності) перед монтажем. Треба створити нову таблицю файлових систем, як ця (рис 1.29):

```
# file system  mount-point  type     options          dump  fsck
#                                                     order
/dev/<xxx>     /          <fff>    defaults          1     1
/dev/<yyy>     swap       swap     pri=1             0     0
proc           /proc      proc     nosuid,noexec,nodev 0   0
sysfs          /sys       sysfs    nosuid,noexec,nodev 0   0
devpts         /dev/pts   devpts   gid=5,mode=620    0     0
tmpfs          /run       tmpfs    defaults          0     0
devtmpfs       /dev       devtmpfs mode=0755,nosuid  0     0
```

Рисунок 1.29 – Зміст файлу /etc/fstab

Де <xxx>, <yyy>і <fff> значення, характерні для системи, наприклад, sda2, sda5, і ext4.

Можна зробити файлову систему ext3 надійною при відключенні живлення для деяких типів жорстких дисків. Для цього треба додати barrier=1 монтування до відповідного запису в /etc/fstab.

Процес складання ядра складається з декількох етапів: настройка, компіляція і установка. Підготую пакет до компіляції виконавши наступну команду (рис 1.30).

make mrproper

Рисунок 1.30 – Зміст файлу /etc/fstab

Виконання цієї команди гарантує, що дерево вихідних кодів ядра абсолютно чисте. Розробники ядра рекомендують, щоб ця команда виконувалася перед кожним процесом компіляції. Зверніть увагу що після розпакування пакету з вихідним кодом не слід покладатися на його "чистоту".

make menuconfig

При бажанні, можна пропустити етап настройки, скопіювавши файл конфігурації ядра .config з вашої хост системи (в тому випадку, якщо такий файл є) в попередньо розпакований каталог ядра linux.

Скомпілюю образ ядра і модулі (рис 1.31):

make

Рисунок 1.31 – команда компіляції

Встановлю модулі, які ядро використовує (рис 1.32):

make modules_install

Рисунок 1.32 – команда компіляції

Після завершення компіляції, необхідно виконати ще кілька кроків. Деякі файли повинні бути скопійовані в каталог /boot.

Шлях до образу ядра залежить від використовуваної платформи. Файл, вказане нижче, може мати довільне найменування, але назва файлу має починатися з vmlinuz для забезпечення сумісності автоматичної настройки процесу завантаження, описаного в наступному розділі. При виконанні наступної команди, буде важати що використовується архітектура x86 (рис 1.33):

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-5.2.8-Linux-9.0
```

Рисунок 1.33 – Команда копіювання

System.map файл, всередині якого знаходиться символьна таблиця адрес функцій і процедур, що використовуються ядром операційної системи Linux. У цій таблиці перераховані імена змінних і функцій та їх адреси в пам'яті комп'ютера. Ця таблиця дуже корисна при налагодженні ядра в разі Kernel panic або Linux oops. System.map генерується при компіляції ядра. Виконаю наступну команду для установки файлу System.map (рис 1.34):

```
cp -iv System.map /boot/System.map-5.2.8
```

Рисунок 1.34 – Команда для установки файлу System.map

Файл конфігурації ядра .config отриманий в результаті настройки make menuconfig містить в собі всі опції конфігурації скомпільованої ядра (рис 1.35):

```
cp -iv .config /boot/config-5.2.8
```

Рисунок 1.35 – Команда копіювання

Встановіть документацію ядра (рис 1.36):

```
install -d /usr/share/doc/linux-5.2.8
cp -r Documentation/* /usr/share/doc/linux-5.2.8
```

Рисунок 1.36 – Команда встановки ядра

Важливо відзначити, що файли в каталозі вихідних кодів ядра не належать користувачеві root. Всякий раз, коли пакет розпаковується від користувача root (Як це і виконувалося всередині середовища chroot), файли мають ті ідентифікатори користувача і групи, які були призначені при розпакуванні.

Зазвичай модулі Linux завантажуються автоматично, але іноді потрібен певний порядок. Програма, яка завантажує модулі, modprobe або insmod, використовує файл /etc/modprobe.d/usb.conf як раз для цієї мети. Цей файл повинен бути створений так, що якщо USB-драйвери (ehci_hcd, ohci_hcd і uhci_hcd) були створені у вигляді модулів, то вони будуть завантажені в необхідному порядку; ehci_hcd повинен бути завантажений до ohci_hcd і uhci_hcd для того, щоб уникнути попереджень під час завантаження.

Треба створити новий файл /etc/modprobe.d/usb.conf виконавши наступну команду (рис 1.37):

```
install -v -m755 -d /etc/modprobe.d
install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true
```

Рисунок 1.37 – Команди встановки

GRUB записує дані на перший фізичний сектор жорсткого диска. Ця область не є частиною будь-якої файлової системи. Програми доступні як модулі GRUB в завантажувальному розділі в каталозі за замовчуванням - /boot/grub/.

Розташування завантажувального розділу - це вибір кожного, його можна налаштувати. Одна рекомендація полягає в тому, щоб мати окремий невеликий (рекомендований розмір - 100 МБ) розділ тільки для завантажувального інформації. Таким чином, кожна збірка, будь то Linux або будь-якої іншої дистрибутив, може звертатися до тих же завантажувальним файлам, і доступ може бути зроблений з будь-завантаженої системи.

Використання поточного розділу Linux буде працювати, але настройка для декількох систем складніше.

Використовуючи наведену вище інформацію, визначте відповідні точки монтування для кореневого розділу (або завантажувального розділу, якщо використовується окремий). У наступному прикладі передбачається, що кореневої (або окремий завантажувальний) розділ є sda2.

Встановлю файли GRUB в каталог /boot/grub і налаштуйте завантажувальний сектор (рис 1.38):

```
grub-install /dev/sda
```

Рисунок 1.38 – Команди для встановлення grub

Створіть файл конфігурації /boot/grub/grub.cfg (рис 1.39):

```
set default=0
set timeout=5
insmod ext2
set root=(hd0,2)
menuentry "GNU/Linux, Linux 5.2.8-Linux-9.0" {
        linux   /boot/vmlinuz-5.2.8-Linux-9.0 root=/dev/sda2 ro
}
```

Рисунок 1.39 – Зміст файлу /boot/grub/grub.cfg

GRUB - надзвичайно потужна програма, і вона забезпечує величезну кількість варіантів завантаження з самих різних пристроїв, що працюють систем і типів розділів. Існує також безліч варіантів настройки таких як графічні екрани заставок, звуки відтворення, введення миші.

1.8 Постановка завдання

Дослідити особливості збірки операційних систем з відкритим кодом. Розглянути розділи диска та файлову систему, пакети і патчі для збірки системи, налаштування робочого оточення, зібрати тимчасову систему, встановити основне програмного забезпечення, конфігурувати систему, зібрати ядро Linux, установка завантажувача.

Проаналізувати вразливостей операційних систем з відкритим кодом;

Продемонструвати можливостей механізмів захисту операційних систем з відкритим кодом та формування рекомендацій. Покращити автентифікацію, встановити політики firewall, встановити служби мережевої безпеки (NSPR NSS), встановити шифрування довільних TCP-з'єднання всередині SSL. Забезпечити Механізми запобігання порушенням програмного забезпечення, встановити шифрування даних. Побудувати репозиторій Linux з використанням механізмів захисти, встановити веб-серверу для репозиторію, вибрати джерела віддзеркалювання репозиторію, написати скрипт автоматичного оновлення репозиторію.

Висновки за розділом 1

У розділі 1 було проаналізовано особливості операційних систем з відкритим кодом. Описується процес створення нових директорій Linux. Це місце, де буде виконуватися компіляція, і куди буде встановлена нова система Linux. Описується які пакети і патчі необхідно завантажити щоб виконати збірку Linux та яким чином їх зберігати на файлову систему. Далі обговорюється процес налаштування робочого оточення. Пояснюється яким чином виконати перший прохід збирання набору інструментів, використовуючи пакети Binutils і GCC. Наступним кроком була інструкція по збірці пакету Glibc - бібліотеки C. Бібліотека C була скомпільована тими засобами, які були встановлені на етапі першого проходу збірки тимчасового набору інструментів. Далі, слід другий прохід збірки тимчасового набору інструментів. На цей раз набір інструментів бу динамічно скомпонований із зібраною на попередньому кроці бібліотекою C. Решта пакети будуть використовувати стан тимчасового набору інструментів після другого проходу.

Команда chroot була використана для зміни кореневого каталогу на створений раніше розділ Linux. Після чого була запущена нова командна оболонка. Ці дії нагадують процес перезавантаження системи із зазначенням ядру що розділ Linux повинен бути кореневим. Система практично не перезавантажується, замість чого використовується команда chroot, тому що для створення завантажувального системи необхідно провести деякі дії, які на даному етапі ще не зроблені. Основною

перевагою є те, що зміна кореневого розділу дозволяє штатно використовувати хост систему, поки відбуватиметься процес складання та конфігурації Linux.

Коли базова система Linux була налаштована, для закінчення установки була налаштування ядра і системний завантажувач.

При наявності файлу etc/Linux-release, можна узнати про встановлену версії Linux. Для цього треба створити такий файл (рис 1.40):

```
echo 9.0 > /etc/lfs-release
```

Рисунок 1.40 – Команди для встановлення файлу

Також непогано створити файл, які покаже статус нової системи по відношенню до стандартів LSB. Щоб створити цей файл (рис 1.41):

```
DISTRIB_ID="Linux"
DISTRIB_RELEASE="0.1"
DISTRIB_CODENAME="<Diplom>"
DISTRIB_DESCRIPTION="Linux"
```

Рисунок 1.41 – Зміст файлу etc/Linux-release

Де в полі 'DISTRIB_CODENAME' унікальна назва нової системи.

По завершенні перезавантаження, система Linux готова до використання.

РОЗДІЛ 2
ОСНОВНІ ВРАЗЛИВОСТІ ОПЕРАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ

2 Місця реєстрації вразливих місць

Все програмне забезпечення має помилки. Іноді помилку можна використати, наприклад, щоб дозволити користувачам отримати розширені привілеї (можливо, отримати кореневу оболонку, або просто отримати доступ до файлів інших користувачів або видалити їх), або дозволити віддаленому веб-сайту збити програму (відмова в обслуговуванні), або за крадіжку даних. Ці помилки позначені як вразливі місця.

Основне місце, де реєструються вразливі місця - cve.mitre.org . На жаль, багато номерів вразливості (CVE-pppp-####) спочатку позначаються лише як "зарезервовані", коли розподіли починають видавати виправлення.

Щоб відстежувати те, що було виявлено, можна стежити за повідомленнями про безпеку одного або кількох дистрибутивів. Наприклад, Debian має захист Debian https://www.debian.org/security/ . Посилання Fedora на безпеку містяться у вікі Fedora https://fedoraproject.org/wiki/Category:Security?rd=Security. Деталі повідомлень про захист Linux від Gentoo обговорюються в Gentoo security https://security.gentoo.org. Нарешті, архіви повідомлень про безпеку Slackware знаходяться в розділі http://www.slackware.com/security.

Найзагальнішим англійським джерелом є повний список розсилки https://seclists.org/fulldisclosure/.

2.1 Порушена автентифікація

Зловмисники можуть отримати доступ до облікових записів користувачів і навіть можуть скомпрометувати всю хост-систему через облікові записи адміністратора, використовуючи вразливості в системах автентифікації. Недоліки автентифікації дозволяють зловмисникові компрометувати паролі, токени сеансів,

ключі автентифікації і можуть бути пов’язані з іншими атаками, що можуть призвести до несанкціонованого доступу до будь-якого іншого облікового запису користувача або сеансу тимчасово, а в деяких випадках і назавжди. Скажімо, у користувача є список слів або словник мільйонів дійсних імен користувачів та паролів, отриманих під час порушення. Він може використовувати їх по одному за надзвичайно менший час, використовуючи автоматизовані інструменти та сценарії в системі входу, щоб перевірити, чи працює хтось. Погана реалізація управління ідентифікацією та контролю доступу призводить до таких вразливостей, як зламана автентифікація.

Додаток вразливий до атак автентифікації, коли дозволяє пробувати різні імена користувачів та паролі, дозволяє атакувати словники або атаки грубої сили без будь-якої стратегії захисту, використовувати прості паролі за замовчуванням або паролі, які просочуються при будь-якому порушенні, виставляє ідентифікатори сеансу в URL, використовує погана схема відновлення пароля, використовує шаблон cookie. Пошкоджену автентифікацію можна легко використати, використовуючи прості інструменти для грубого примусу та атаки на словники з хорошим словником. Такі типи атак можна запобігти за допомогою багатофакторних систем автентифікації, здійснивши слабкі перевірки паролів, запустивши пароль через базу даних помилкових паролів, не використовуючи облікові дані за замовчуванням, узгоджуючи політику складності паролів, використовуючи хорошу сторону сервера менеджер сеансів, який генерує новий випадковий ідентифікатор сеансу після входу тощо.

Пошкоджена вразливість автентифікації може призвести до компрометації кількох облікових записів користувачів та облікового запису адміністратора, тобто все, що потрібно зловмисникові, щоб зламати систему. Такі напади призводять до викрадення особистих даних, шахрайства із соціальним забезпеченням, відмивання грошей та розголошення секретної інформації. Атаки включають атаки на словники, примушування до грубих дій, викрадення сесій та атаки на управління сеансами.

2.2 Вплив чутливих даних

Іноді веб-програми не захищають конфіденційні дані та інформацію, такі як паролі, облікові дані бази даних тощо. Зловмисник може легко викрасти або змінити ці слабо захищені облікові дані та використовувати їх у незаконних цілях. Конфіденційні дані повинні шифруватися, коли вони перебувають у стані спокою або в дорозі, і мати додатковий рівень захисту, інакше зловмисники можуть їх викрасти. Зловмисники можуть отримати в руки конфіденційні відкриті дані та викрасти хешовані або очистити текстові дані користувачів та облікові дані бази даних із сервера або веб-браузера. Наприклад, якщо база даних паролів використовує несолоні або прості хеші для зберігання паролів, недолік завантаження файлу може дозволити зловмисникові отримати базу даних паролів, що призведе до виявлення всіх паролів за допомогою райдужної таблиці заздалегідь розрахованих хешів.

Основна вада полягає не тільки в тому, що дані не зашифровані, навіть якщо вони зашифровані, але слабке генерування ключів, слабкі алгоритми хешування, слабке використання шифру також можуть призвести до цих типів однієї з найпоширеніших атак. Для запобігання цим типам атак спочатку класифікуйте, який тип даних можна вважати конфіденційним згідно із законодавством про конфіденційність, та застосуйте засоби контролю відповідно до класифікації. Для передачі даних зашифруйте їх із захищеними протоколами, тобто TLS із шифрами PFS тощо.

Такі типи вразливостей можуть призвести до виявлення надзвичайно конфіденційної інформації, таких як дані кредитної картки, медичні картки, паролі та будь-які інші особисті дані, які можуть призвести до крадіжки особистих даних та банківського шахрайства тощо.

2.3 Порушений контроль доступу

Контроль доступу надає користувачам привілеї для виконання конкретних завдань. Порушена вразливість контролю доступу має місце, коли користувачі не

мають належних обмежень щодо завдань, які вони можуть виконувати. Зловмисники можуть використати цю вразливість, яка в кінцевому підсумку може отримати доступ до несанкціонованих функцій або інформації. Скажімо, веб-програма дозволяє користувачеві змінити обліковий запис, з якого він увійшов, просто змінивши URL-адресу на рахунок іншого користувача без подальшої перевірки. Використання вразливості контролю доступу - це атака будь-якого зловмисника, яку можна знайти вручну, а також за допомогою інструментів SAFT та DAFT. Ці вразливості існують через відсутність тестування та автоматизованого виявлення веб-програм, хоча найкращий спосіб їх знайти - зробити це вручну.

Уразливості містять ескалацію привілеїв, тобто діяти як користувач, яким ви не є, або діяти як адміністратор, поки ви користувач, обходячи перевірку контролю доступу, просто змінюючи URL-адресу або змінюючи стан програми, маніпулювання метаданими, дозволяючи первинному ключу змінюватись як інший первинний ключ користувача тощо. Щоб запобігти подібним атакам, механізми контролю доступу повинні бути реалізовані в коді на стороні сервера, де зловмисники не можуть змінити засоби контролю доступу. Для пом'якшення подібних видів атак необхідно забезпечити дотримання унікальних обмежень на бізнес-програми моделями доменів, відключення переліку каталогів серверів, попередження адміністратора про повторні невдалі спроби входу в систему, недійсність токенів JWT після виходу з системи.

Зловмисники можуть виступати в ролі іншого користувача або адміністратора, використовуючи цю вразливість для виконання шкідливих завдань, таких як створення, видалення та модифікація записів тощо. Можлива велика втрата даних, якщо дані не захищені навіть після порушення.

2.4 Неправильна конфігурація безпеки

Найбільш поширеною вразливістю є неправильне налаштування безпеки. Основною причиною уразливості є використання конфігурації за замовчуванням, неповна конфігурація, конфігурації Adhoc, погано налаштовані заголовки HTTP та детальні повідомлення про помилки, що містять більше інформації, ніж користувач

насправді повинен був знати. На будь-якому рівні веб-програми можуть статися помилкові конфігурації безпеки, тобто база даних, веб-сервер, сервер додатків, мережеві служби тощо. Зловмисники можуть використовувати нерозпаковані системи або отримувати доступ до незахищених файлів і каталогів, щоб несанкціоновано затримати систему. Наприклад, надмірно розгорнуте повідомлення про помилку програми, яке допомагає зловмисникові дізнатися про вразливі місця в системі додатків та спосіб роботи. Для виявлення таких типів недоліків безпеки можна використовувати автоматизовані інструменти та сканери.

Веб-програма містить уразливість цього типу, якщо в ній відсутні заходи щодо посилення безпеки в будь-якій частині програми, непотрібні порти відкриті або це дозволяє зайві функції, використовуються паролі за замовчуванням, обробка помилок виявляє інформативні помилки зловмисникові не виправлене або застаріле програмне забезпечення безпеки тощо. Цього можна запобігти, видаливши непотрібні функції коду, тобто мінімальну платформу без зайвих функцій, документації тощо, дозволивши завдання оновити та виправити діри в безпеці як частину процесів управління виправленнями, використання процесу для перевірки ефективності вжитих заходів безпеки, використання повторюваного процесу зміцнення, щоб полегшити розгортання іншого середовища, яке належним чином заблоковано.

Ці типи вразливостей або недоліків дозволяють зловмиснику отримати несанкціонований доступ до системних даних, що призводить до повної компрометації системи.

2.5 Використання компонентів з відомими вразливими місцями

Більшість розробників у веб-програмі використовують різні компоненти, такі як бібліотеки, фреймворки та програмні модулі. Ці бібліотеки допомагають розробнику уникнути зайвої роботи та забезпечують необхідну функціональність. Зловмисники шукають недоліки та вразливості в цих компонентах для координації атаки. У разі виявлення лазівки в безпеці в компоненті можна зробити всі сайти, що використовують один і той же компонент, уразливими. Експлойти цих уразливостей

вже доступні, тоді як написання власного експлоїту з нуля вимагає багато зусиль. Це дуже поширена і широко поширена проблема, використання великої кількості компонентів при розробці веб-додатків може призвести до того, що навіть не знати і не розуміти всіх використовуваних компонентів, виправлення та оновлення всіх компонентів - це довгий шлях.

Додаток вразливий, якщо розробник не знає версії використовуваного компонента, програмне забезпечення застаріле, тобто операційна система, СУБД, програмне забезпечення, середовища виконання та бібліотеки, сканування вразливості не проводиться регулярно, сумісність виправлених програмне забезпечення не перевіряється розробниками. Цього можна запобігти, видаляючи невикористані залежності, файли, документацію та бібліотеки, регулярно перевіряючи версію клієнтських та серверних компонентів, отримуючи компоненти та бібліотеки з офіційних та надійних захищених джерел, контролюючи невідпаковані бібліотеки та компоненти, забезпечуючи план для регулярного оновлення та виправлення вразливих компонентів.

Ці вразливості призводять до незначних наслідків, але також можуть призвести до компрометації сервера та системи. Багато великих порушень спиралися на відомі вразливості компонентів. Використання вразливих компонентів підриває захист додатків і може стати відправною точкою для великої атаки.

Висновки за розділом 2

У розділі 2 було розглянуто основні вразливості для операційних систем з відкритим кодом. Спочатку було проаналізовано основне місце, де реєструються вразливі місця для unix-подібних операційних систем. Вразливі місця впливають на всі сутності, пов'язані з операційними системами. Про ці вразливості слід подбати, щоб забезпечити безпечне середовище для користувачів. Зловмисники можуть використовувати ці вразливості для компрометації системи, отримання її та посилення привілеїв. Вплив скомпрометованої веб-програми може бути візуалізований від викрадених облікових даних кредитної картки та викрадення

особистих даних до витоку дуже конфіденційної інформації тощо залежно від потреб та векторів атак зловмисних осіб.

РОЗДІЛ 3
МЕХАНІЗМИ ЗАХИСТУ ОПЕРАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ

3 Механізми захисту доступності

Доступ для користувачів зазвичай здійснюється за допомогою логіна або програми, призначеної для роботи з функцією входу. У цьому розділі покажу, як покращити вхід, встановлюючи політики за допомогою модулів PAM . Доступ через мережі також може бути забезпечений політиками, встановленими iptables, які зазвичай називають брандмауером. Бібліотеки служб мережевої безпеки (NSS) та Netscape Portable Runtime (NSPR) можна встановити та надати спільний доступ до багатьох програм, що потребують їх використання. Для програм, які не забезпечують найкращого захисту можна використовувати пакет Stunnel, щоб обернути демон програми всередині тунелю SSL.

3.1 Покращення входу

Пакет Linux PAM містить підключаються модулі автентифікації, які використовуються для того, щоб місцевий системний адміністратор міг вибрати спосіб автентифікації програм.

Налаштування файлів /etc/security/* і /etc/pam.d/*.

Інформація про конфігурацію розміщується в /etc/pam.d/. Нижче наведено приклад файлу (рис 3.1):

```
auth        required      pam_unix.so    nullok
account     required      pam_unix.so
session     required      pam_unix.so
password    required      pam_unix.so    nullok
```

Рисунок 3.1 – Приклад файлу /etc/pam.d/

Тепер налаштуємо кілька загальних файлів. Як root користувач (рис 3.2):

```
install -vdm755 /etc/pam.d &&
account   required    pam_unix.so
auth      required    pam_unix.so
session   required    pam_unix.so
password  required    pam_unix.so       sha512 shadow try_first_pass
```

Рисунок 3.2 – Команди налаштування Linux PAM

Тепер додайте обмежувальний /etc/pam.d/other файл конфігурації. За допомогою цього файлу програми, про які відомо PAM, не запускатимуться, якщо не буде створений файл конфігурації спеціально для цієї програми (рис 3.3).

```
auth        required        pam_warn.so
auth        required        pam_deny.so
account     required        pam_warn.so
account     required        pam_deny.so
password    required        pam_warn.so
password    required        pam_deny.so
session     required        pam_warn.so
session     required        pam_deny.so
```

Рисунок 3.3 – Команди налаштування Linux PAM

3.1.1 Встановлення політик firewall

iptables - це програма командного рядка користувальницького простору, яка використовується для налаштування набору правил фільтрації пакетів ядра Linux 2.4 та пізніших версій.

Персональний брандмауер створений, щоб дозволити отримати доступ до всіх послуг, що пропонуються в Інтернеті, зберігаючи при цьому ваш комп'ютер у безпеці, а ваші дані конфіденційними.

Нижче наведено трохи модифіковану версію рекомендації Расті Рассела з фільтрування пакетів Linux 2.4 HOWTO . Він все ще застосовується до ядер Linux 5.x.

Файл /etc/rc.d/rc.iptables:

Увімкнути захист від eho-сигналу (рис 3.4):

```
echo 1> / proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Рисунок 3.4 – Команда для включення захисту від exo-сигналу

Вимкнути вихідні маршрутизовані пакети (рис 3.5):

```
echo 0> /proc/sys/net/ipv4/conf/all/accept_source_route
echo 0> /proc/sys/net/ipv4/conf/default/accept_source_route
```

Рисунок 3.5 – Команда для вимкнення вихідних маршутизованих пакетів

Увімкніть захист файлів cookie TCP SYN (рис 3.6):

```
echo 1>/proc/sys/net/ipv4/tcp_syncookies
```

Рисунок 3.6 – Команда для включення захисту файлів cookie

Вимкнути переадресацію ICMP (рис 3.7):

```
echo 0> /proc/sys/net/ipv4/conf/default/accept_redirects
```

Рисунок 3.7 – Команда для включення переадресації ICMP

Не надсилати повідомлення про переадресацію (рис 3.8):

```
echo 0> /proc/sys/net/ipv4/conf/default/send_redirects
echo 0> /proc/sys/net/ipv4/conf/default/send_redirects
```

Рисунок 3.8 – Команда щоб не надсилати повідомлення про переадресацію

Видалить підроблені пакети, що надходяться на інтерфейсі, де призведе до того, що відповідь виведе інший інтерфейс (рис 3.9):

```
echo 1> /proc/sys/net/ipv4/conf/all/rp_filter
echo 1> /proc/sys/net/ipv4/conf/default/rp_filter
```

Рисунок 3.9 – Команда для щоби вдалити підроблені пакети

Дозволити лише локальні підключення (рис 3.10):

```
iptables -A INPUT -i lo -j ACCEPT
chmod 700 /etc/rc.d/rc.iptables
```

Рисунок 3.10 – Команда шоб Дозволити лише локальні підключення

3.1.2 Встановлення служб мережевої безпеки (NSPR NSS)

Netscape Portable Runtime (NSPR) забезпечує нейтральний для платформи API для системного рівня та подібних до libc функцій.

Пакет Network Security Services (NSS) - це набір бібліотек, призначених для підтримки крос-платформної розробки клієнтських та серверних додатків із підтримкою безпеки.

Якщо встановлено p11-kit-0.23.22, модуль довіри p11-kit (/usr/lib/pkcs11/p11-kit-trust.so) можна використовувати як заміну, що /usr/lib/libnssckbi.so відкривається, для прозорого надання системних ЦС доступним програмам, що обізнані з NSS, а не як статичний список, наданий /usr/lib/libnssckbi.so. Як root користувач виконайте такі команди (рис 3.11):

```
ln -sfv ./pkcs11/p11-kit-trust.so /usr/lib/libnssckbi.so
```

Рисунок 3.11 – Команда для створення посилання

Крім того, для залежних додатків, які не використовують внутрішню базу даних (/usr/lib/libnssckbi.so), /usr/sbin/make, включений на сторінку make-ca-1.7, може генерувати загальносистемну базу даних NSS за допомогою -n комутатора або шляхом модифікації /etc/make-ca.conf файлу.

3.1.3 Шифрування довільних TCP-з'єднання всередині SSL

Пакет stunnel містить програму, яка дозволяє шифрувати довільні TCP-з'єднання всередині SSL (рівень захищених сокетів), щоб можна було легко спілкуватися з клієнтами через захищені канали. stunnel можна використовувати для додавання функцій SSL до часто використовуваних демонів Inetd, таких як сервери POP-2, POP-3 та IMAP, а також до окремих демонів, таких як NNTP, SMTP та HTTP.

Налаштування файлів /etc/stunnel/stunnel.conf

Як root користувач створіть каталог, який використовується для .pid файлу, створеного при запуску демона stunnel (рис 3.12):

```
install -v -m750 -o stunnel -g stunnel -d /var/lib/stunnel/run &&
chown stunnel:stunnel /var/lib/stunnel
```

Рисунок 3.12 – Команда для встановки та для зміни власника

Далі створіть базовий /etc/stunnel/stunnel.conf файл конфігурації, використовуючи такі команди як root користувача (рис 3.13):

```
pid    = /run/stunnel.pid
chroot = /var/lib/stunnel
client = no
setuid = stunnel
setgid = stunnel
cert   = /etc/stunnel/stunnel.pem
```

Рисунок 3.13 – Зміст файлу /etc/stunnel/stunnel.conf

```
;debug = 7
;output = stunnel.log
;[https]
;accept  = 443
;connect = 80
;; "TIMEOUTclose = 0" is a workaround for a design flaw in Microsoft SSL
;; Microsoft implementations do not use SSL close-notify alert and thus
;; they are vulnerable to truncation attacks
;TIMEOUTclose = 0
```

Рисунок 3.13 – Зміст файлу /etc/stunnel/stunnel.conf

Нарешті, додайте службу (служби), яку хочете зашифрувати, до файлу конфігурації. Формат такий:

[<service>]

accept = <hostname:portnumber>

connect = <hostname:portnumber>

Якщо використовується stunnel для шифрування демона, запущеного з [x] inetd треба вимкнути цей демон у /etc/[x]inetd.conf файлі та увімкнути відповідну <service>службу _stunnel. Треба додати відповідний запис /etc/services.

3.2 Механізми запобігання порушенням програмного забезпечення

Запобігання порушенням, як троянські програми, допомагають такі програми, як GnuPG, зокрема можливість підтвердження підписаних пакетів, що розпізнає модифікації tarball після створення пакувальника.

3.2.1 Шифрування даних GnuPG

GnuPG - це повна та безкоштовна реалізація стандарту OpenPGP, визначеного RFC4880 (також відомий як PGP). GnuPG дозволяє шифрувати та підписувати ваші

дані та комунікації; він має універсальну систему управління ключами, а також модулі доступу для всіх видів каталогів відкритих ключів. GnuPG, також відомий як GPG, є інструментом командного рядка з функціями для легкої інтеграції з іншими програмами. Доступні численні інтерфейсні програми та бібліотеки. GnuPG також забезпечує підтримку S/MIME і Secure Shell (ssh).

3.3 Побудова репозиторія Linux з використанням механізмів захисту

Що б поліпшити безпеку і прискорити процес — можна створити окремий локальний депозитарій на одній машині, яка буде оновлювати в ньому пакети, і потім роздавати оновлення іншим машинам в мережі. Буде розглянуто конфігурації apt та структуру репозиторія apt. Репозиторій пакетів Debian в основному використовуються для автоматичного зберігання та отримання пакетів. Менеджери пакетів використовують lib apt для вилучення пакетів із зовнішніх носіїв і з Інтернету.

На сторінці sources.list вказаний такий формат вихідного коду пакета:

deb url distribution [component1] [component2] [...]

Приклад:

```
deb https://debian.volia.net/debian/ stable main contrib non-free
deb-src https://debian.volia.net/debian/ stable main contrib non-free
```

Рисунок 3.11 – Команда для створення посилання

Тут deb вказує, що це джерело для бінарних пакетів, deb-src - для пакетів з вихідним кодом. В архіві можуть бути або пакети з вихідним кодом, або бінарні пакети, або обидва, але вони повинні бути вказані окремо для apt.

URI, в цьому випадку https://debian.volia.net/debian/ вказує корінь репозиторії. Часто архіви Debian знаходяться в каталозі debian/ на сервері, але можуть бути де завгодно(наприклад, багато дзеркал зберігають його в каталозі pub/linux/debian).

Частина дистрибутива (в такому випадку стабільна stable) вказує підкаталог в $ARCHIVE_ROOT/dists. Він може містити додаткові косі риси для вказівки вкладених підкаталогів, наприклад stable/updates. Дистрибутив зазвичай відповідає набору або кодового імені, вказаного в файлах випуску.

Щоб завантажити пакети зі сховища, apt завантажить файл InRelease або Release з каталогу $ARCHIVE_ROOT/dists/$DISTRIBUTION.

Файли InRelease підписуються в рядку, в той час, як файли Release повинні мати супутній файл Release.gpg.

У файлі Release перераховані індексні файли для дистрибутива і їх хеші (індексний файл вказано щодо розташування файлу Release).

Для завантаження індексу основного компонента apt буде сканувати файл Release на наявність хешів файлів в головному каталозі.

Індекси бінарних пакетів знаходяться в підкаталозі binary-$arch каталогів компонентів.

$ARCHIVE_ROOT/dists/$DISTRIBUTION/$COMPONENT/binary-$arch/Packages.xz

Приклад:

https://debian.volia.net/debian/dists/buster/main/binary-amd64/Packages.xz

Вихідні індекси знаходяться в підкаталозі source.

Індекси пакетів перераховують конкретні вихідні або двійкові пакети щодо кореня архіву.

Щоб уникнути дублювання файлів, двійкові та вихідні пакети зазвичай зберігаються в підкаталозі пулу кореневого архіву. Однак індекси пакетів і джерел можуть перераховувати будь-який шлях щодо кореня архіву. Передбачається, що пакети розміщуються в підкаталозі кореня архіву, відмінному від dists, а не безпосередньо в корені архіву. Розміщення пакетів безпосередньо в корені архіву не тестується, і деякі інструменти можуть не індексувати або витягувати розміщені там пакети.

Вміст і індекси перекладу не залежать від архітектури й поміщаються в каталог dists/$DISTRIBUTION/$COMPONENT, а не в підкаталог architecture.

Типи офіційних депозитаріїв в Debian можна подивитися в:

$ARCHIVE_ROOT/dists/README

На клієнті файли InRelease і Packages зберігаються в var/lib/apt/lists/

Головна сторінка дзеркала перераховує архіви, доступні для віддзеркалення.

Користувачі будуть шукати архів debian/ для установки Debian через мережу, для створення дисків (за допомогою jigdo), і для поновлення вже встановлених систем.

debian-cd/ - це архів, який не однаковий для різних серверів дзеркал. На деяких сайтах він містить шаблони jigdo для створення образів дисків (використовується спільно з файлами з debian /), на деяких він містить вже створені образи дисків, а на деяких сайтах міститься обидва варіанти. Детальну інформацію про створення дзеркал дивіться на сторінці віддзеркалення образів дисків.

debian-archive/ містить справжній архів старих які вийшли з ужитку версій Debian. Головним чином він буде цікавий лише малому числу користувачів.

Архів debian-security/ містить оновлення безпеки, випущені командою безпеки Debian. Хоча це здається цікавим для кожного, але документація не рекомендує користувачам використовувати дзеркала для отримання оновлень безпеки, а навіть навпаки просить їх завантажувати оновлення безпосередньо з нашої розподіленої служби security.debian.org.

Інформацію про розміри дзеркала дивіться на сторінці розмір дзеркала:

https://www.debian.org/mirror/size/

3.3.1 Установка веб-серверу для репозиторію

Apache доступний у сховищах програмного забезпечення CentOS, що використовуються за замовчуванням, тобто можно встановити його за допомогою менеджера пакетів yum.

За допомогою користувача sudo без прав root, налаштованого згідно з попередніми вимогами, оновити локальний Індекс пакета Apache httpd, щоб показати останні зміни (рис 3.14):

sudo yum update httpd

sudo yum update httpd

Рисунок 3.14 – Команда для оновлення локальних індексів yum

Після оновлення пакетів встановити пакет Apache (рис 3.15):

sudo yum install httpd

Рисунок 3.15 – Команда для встановки Apache

Після підтвердження установки yum виконає установку Apache і всіх необхідних залежностей.

Потрібно відкрити порт 80, щоб дати Apache можливість обслуговувати запити через HTTP. Для цього можна активувати в брандмауері firewall службу http за допомогою наступної команди (рис 3.15):

sudo firewall-cmd --permanent --add-service=http

Рисунок 3.15 – Команда щоб активувати firewall службу http

Включити службу https (рис 3.16):

sudo firewall-cmd --permanent --add-service=https

Рисунок 3.16 – Команда для включення служби ssl

Потім перезавантажу брандмауер, щоб нові правила почали діяти (рис 3.17):

sudo firewall-cmd --reload

Рисунок 3.17 – Команда для налаштування брандмауеру

Apache не запускається автоматично на CentOS після завершення установки. Це потрібно буде запустити процес Apache вручну (рис 3.18):

```
sudo systemctl start httpd
```

Рисунок 3.18 – Команда для запуску процесу

Переконатися можна, що служба запущена за допомогою наступної команди:

```
sudo systemctl status httpd
```

```
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
  Active: active (running) since Mon 2021-04-12 12:21:19 EEST; 3h 12min ago
    Docs: man:httpd.service(8)
Main PID: 17775 (httpd)
  Status: "Total requests: 15; Idle/Busy workers 100/0;Requests/sec: 0.0013; Bytes
served/sec:    1 B/sec"
  Tasks: 278 (limit: 11100)
  Memory: 22.9M
  CGroup: /system.slice/httpd.service
          ├─17775 /usr/sbin/httpd -DFOREGROUND
          ├─17776 /usr/sbin/httpd -DFOREGROUND
          ├─17777 /usr/sbin/httpd -DFOREGROUND
          ├─17778 /usr/sbin/httpd -DFOREGROUND
          ├─17779 /usr/sbin/httpd -DFOREGROUND
          └─18004 /usr/sbin/httpd -DFOREGROUND
```

Рисунок 3.19 – Работа Apache

Як видно з результатів, служба успішно запущена. Однак найкраще протестувати її запуск за допомогою запиту сторінки з Apache.

Можно відкрити сторінку Apache за замовчуванням, щоб підтвердити роботу програмного забезпечення через вашу IP-адресу.

Створю папку $ARCHIVE_ROOT де буде зберігається репозиторій (рис 3.20):

```
mkdir /var/www/html/$ARCHIVE_ROOT
```

Рисунок 3.20 – Работа для створення папки

3.3.2 Джерела віддзеркалювання

ftp.debian.org не є канонічним місцем розміщення пакетів Debian, це лише один з кількох серверів, які оновлюються з внутрішнього сервера Debian. Існує безліч відкритих дзеркал, що підтримують rsync, які цілком підходять в якості джерела для віддзеркалення. Використовуйте близький до вас (в сенсі його мережевого розміщення) сервер.

Не слід використовувати в якості джерела зеркалирования ім'я сервісу, які дозволяється в кілька адрес (наприклад, ftp.us.debian.org), оскільки порушення синхронізації між такими дзеркалами може привести до синхронізації між різними станами вашого власного процесу синхронізації дзеркала. Крім того, зауважте, що представники Debian гарантує роботу тільки HTTP на ftp.країна.debian.org. Щоб виконувати віддзеркалення за використанням rsync (рекомендується використовувати ftpsync), то виберіть відповідне ім'я сайту для машини, що надає в даний час ftp.країна.debian.org. (Для цього слід звернутися до каталогу на цьому сервері /debian/project/trace).

3.3.3 Віддзеркалювання репозиторію

Рекомендованим методом віддзеркалення є набір сценаріїв ftpsync, який доступний в двох видах:

tar-архів https://ftp-master.debian.org/ftpsync.tar.gz

з git-репозиторію (рис 3.21):

```
git clone https://salsa.debian.org/mirror-team/archvsync.git
```

Рисунок 3.21 – Команда для клонування git

Як протокол віддзеркалювання документація рекомендує використовувати rsync .

Щоб дзеркало було доступно тільки у вашій мережі, або потрібні тільки певні набори пакетів (наприклад, тільки buster main amd64 випуск). Тоді потрібно синхронізувати:

$ARCHIVE_ROOT/dists/buster/main/binary-amd64/

$ARCHIVE_ROOT/dists/buster/InRelease

$ARCHIVE_ROOT/dists/buster/Release

$ARCHIVE_ROOT/dists/buster/Release.gpg

У файлі $ARCHIVE_ROOT/dists/buster/main/binary-amd64/Packages.xz оберіть потрібні пакети. Там структура файл у форматі RFC # 733 ARPA INTERNET TEXT MESSAGES. Для більш зручної обробки можна перевести в json за допомогою модуля debian-parser для Python (рис 3.22):

```
from debian_parser import PackagesParser
sample_pacakges_file = open("Packages", "r").read()
parser = PackagesParser(sample_pacakges_file)
data = parser.parse()
f = open('parser.txt', 'w')
f.write(str(data))
for row in data:
  f.write(str(row))
```

Рисунок 3.22 – Скрипт debian-parser

```
dirs=("pool/*")
for dir in ${dirs[*]}
do
  wget -r -nH --cut-dirs=1 --no-parent --reject="index.html*" -P /storage/www/debian/
https://debian.volia.net/debian/$dir
done
```

Рисунок 3.23 – Скрипт синхронізації з wget

```
        dirs=("pool/*")
for dir in ${dirs[*]}
do
  rsync -avPR rsync://debian.volia.net/debian/$dir /storage/www/debian/
done
```

Рисунок 3.24 – Скрипт синхронізації з rsync

Але документація говорить не використовувати власні сценарії і не використовуйте rsync в режимі одного проходу. Використання ftpsync гарантує, що поновлення здійснюються так, що apt працює нормально. Зокрема, ftpsync обробляє перекази, вміст та інші файли метаданих, щоб при роботі apt не виникли помилки при виконанні перевірок у разі, коли користувач оновлює список пакетів під час оновлення дзеркала. Більш того, цей інструмент також створює trace-файли, що містять додаткову інформацію, яка корисна для визначення того, працює дзеркало чи ні, які архітектури на ньому доступні, а також звідки воно оновлюється.

Беручи до уваги великий розмір архіву Debian, можна порадити віддзеркалювати лише частина архіву. Відкриті дзеркала повинні містити всі випуски (тестований, нестабільний і т. Д.), Але можна обмежити набір архітектури. У файлі настройки ftpsync є опції ARCH_EXCLUDE і ARCH_INCLUDE для цієї мети.

Також можна використовувати готовий сценарій debmirror. Приклад використання debmirror:

```
debmirror --progress --nosource --passive --host=debian.volia.net --method=ftp --root=debian --dist=buster,buster-updates --section=main --arch=amd64 --exclude="games" --ignore-release-gpg /storage/www/mrepo/debian/
```

де --progress - відображати індикатор виконання при скачуванні файлів.

--nosource - не вмикати в копію архіви з вихідними текстами.

--passive - завантажувати в пасивному режимі.

--host - вказує віддалений вузол, з якого потрібно завантажувати файли. За замовчуванням використовується вузол ftp.debian.org, але дуже вітається використання найближчого дзеркала.

--method - вказує спосіб скачування файлів. В даний час підтримуються наступні способи: ftp, hftp (ftp через http-проксі), http або rsync. Щоб з'єднатися з сервером rsync потрібно помістити перед кореневим каталогом символ ':' (наприклад,: debian, що означає вузол :: debian).

--root - вказує каталог на віддаленому вузлі, який є кореневим для архіву Debian. За замовчуванням використовується каталог / debian, який підходить для більшості дзеркал. У кореневому каталозі знаходиться підкаталог dists /.

--dist - вказати одну збірку (sarge, etch, lenny, sid) Debian для копіювання. Цей ключ може використовуватися кілька разів, а в одній опції може бути зазначено кілька дистрибутивів, розділених комами. Використання посилань (stable, testing, unstable) може привести до несподіваних результатів, але можно додати ці посилання вручну. За замовчуванням копіюється sid.

--section - вказує розділ Debian для копіювання. За замовчуванням копіюються розділи main, contrib, non-free, main / debian-installer.

--arch - вказує архітектури процесорів, для яких потрібно завантажувати виконавчі пакети. За замовчуванням використовується опція --arch = i386. При вказівці --arch = none архітектурно-залежні виконавчі пакети скачиваться не будуть.

--exclude= - ніколи не завантажувати файли, чиї імена збігаються з зазначеним регулярним виразом. Опція може бути вказана кілька разів.

--ignore-release-gpg - чи не завершити роботу з помилкою, якщо відсутній файл Release.gpg.

Якщо потрібно віддзеркалювати декілька репозиторіїв то потрібно зробити конфігурацію в /etc/debmirror.conf.

Та запускати окремо (рис 3.25):

debmirror --nosource --host=debian.volia.net --root=debian --dist=buster,buster-updates /storage/www/mrepo/debian/

debmirror --nosource --host=security.debian.org --root=debian-security --dist=buster/updates /storage/www/mrepo/debian/

Рисунок 3.25 – Команда для запуску debmirror

3.3.4 Автоматичне оновлення репозиторію

Головний архів оновлюється чотири рази на день. Дзеркала зазвичай починають оновлюватися близько 3:00, 9:00, 15:00 і 21:00 (весь час по UTC), але це не фіксований час, і не треба спиратися на ці часи при створення дзеркал.

Дзеркало має оновлюватися через кілька годин після початку поновлення головного дзеркала. Треба перевірити, чи залишив сайт, з якого йде віддзеркалення, файл з відміткою часу в його піддиректорії project/trace/. Файл з відміткою часу буде названий як сайт, і він буде містити повне час останнього оновлення його дзеркала. Треба додати пару годин до цього часу (для впевненості) і потім Віддзеркалюються.

Важливо, щоб дзеркало було синхронізовано з основним архівом . Як мінімум 4 оновлення протягом 24 годин потрібні для того, щоб гарантувати, що дзеркало є дійсним відображенням архіву. Пам'ятайте, дзеркала, які не синхронізовані з основним архівом, що не будуть вказані в списку офіційних дзеркал.

Найлегший шлях автоматично щодня запускати віддзеркалення, це використовувати cron (рис 3.26):

crontab -e

*0 0 * * * deb_repo_update.sh*

Рисунок 3.26 – Команда для створення посилання

Висновки за розділом 3

У розділі 3 було були розроблені механізми захисту операційних систем з відкритим кодом. Доступ для користувачів зазвичай здійснюється за допомогою логіна. Було розглянуто як покращити вхід, встановлюючи політики за допомогою модулів PAM. Доступ до мережі також може бути забезпечений політиками, встановленими фаєврovла. Бібліотеки служб мережевої безпеки (NSS) та Netscape Portable Runtime (NSPR) встановив та надав спільний доступ до багатьох програм, що потребують їх використання. Stunnel, щоб обернути демон програми всередині тунелю SSL. Запобігання порушенням, як троянські програми, допомагають такі програми, як GnuPG, зокрема можливість підтвердження підписаних пакетів, що розпізнає модифікації tarball після створення пакувальника.

В ході роботи я зіткнувся з тим що серевер в корпоративних мережах не можуть оновлюватися. Це може бути проблемою безпеки. З цього я розробив репозиторій по оновленню дистребутивів. Їх важливих пакетів і безпеки. Розглянув структур директорій сховища, установка веб-сервера, джерела віддзеркалювання, як правильно віддзеркалювати репозіторій та як зробити автоматичне оновлення репозіторію.

ВИСНОВКИ

Для досягнення поставленої мети дослідив особливості збірки операційних систем з відкритим кодом. Розглянув розділи диска та файлову систему, пакети і патчі для збірки системи, налаштував робоче оточення, зібрав тимчасову систему, встановив основне програмного забезпечення, конфігурував систему, зібрав ядро Linux, налаштував завантажувач.

Проаналізувати вразливостей операційних систем з відкритим кодом;

Продемонстрував можливості механізмів захисту операційних систем з відкритим кодом та формував рекомендації. Покращив автентифікацію, встановив політики firewall, встановив служби мережевої безпеки (NSPR NSS), встановив шифрування довільних TCP-з'єднання всередині SSL. Забезпечив механізми запобігання порушенням програмного забезпечення, встановити шифрування даних. Побудував репозиторій Linux з використанням механізмів захисти, встановив веб-сервер для репозиторію, вибрав джерела віддзеркалювання, написав скрипт автоматичного оновлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Linux from scratch – version 10.1 Chapter 3 Packages and Patches [Електронний ресурс]. – Режим доступу: https://www.linuxfromscratch.org/lfs/view/stable/chapter03/packages.html

2. Linux from scratch – version 10.1 Chapter 3 Packages and Patches [Електронний ресурс]. – Режим доступу: https://www.linuxfromscratch.org/lfs/view/stable/chapter03/patches.html

3. Building and Installing Software Packages for Linux [Електронний ресурс]. – Режим доступу: http://www.tldp.org/HOWTO/Software-Building-HOWTO.html

4. Beginner's Guide to Installing from Source [Електронний ресурс]. – Режим доступу: http://moi.vonos.net/linux/beginners-installing-from-source/

5. The Open Group Base Specifications Issue [Електронний ресурс]. – Режим доступу: http://pubs.opengroup.org/onlinepubs/9699919799/

6. Filesystem Hierarchy Standard Version 3.0 [Електронний ресурс]. – Режим доступу: http://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

7. Офіціальна сторінка Attr [Електронний ресурс]. – Режим доступу: https://savannah.nongnu.org/projects/attr

8. Сторінка программ GNU [Електронний ресурс]. – Режим доступу: http://www.gnu.org/software/software.html

9. An implementation of Unix dc and POSIX bc with GNU and BSD extensions. Finished, but well-maintained [Електронний ресурс]. – Режим доступу: https://git.yzena.com/gavin/bc

10. E2fsprogs: Ext2/3/4 Filesystem Utilities [Електронний ресурс]. – Режим доступу: http://e2fsprogs.sourceforge.net/

11. Elfutils Utilities [Електронний ресурс]. – Режим доступу: https://sourceware.org/ftp/elfutils/

12. Expat xml parser [Електронний ресурс]. – Режим доступу: https://libexpat.github.io/

13. Expat utilities [Електронний ресурс]. – Режим доступу: https://core.tcl.tk/expect/

14. Fine Free File Command [Електронний ресурс]. – Режим доступу: https://www.darwinsys.com/file/

15. GitHub flex [Електронний ресурс]. – Режим доступу: https://github.com/westes/flex

16. Офіціальна сторінка gawk [Електронний ресурс]. – Режим доступу: http://www.gnu.org/software/gawk/

17. GCC, the GNU Compiler Collection [Електронний ресурс]. – Режим доступу: https://gcc.gnu.org/

18. Internet Assigned Numbers Authority (IANA) [Електронний ресурс]. – Режим доступу: https://www.iana.org/protocols/

19. Офіціальна сторінка intltool [Електронний ресурс]. – Режим доступу: https://freedesktop.org/wiki/Software/intltool

20. Офіціальна сторінка iproute2 [Електронний ресурс]. – Режим доступу: https://www.kernel.org/pub/linux/utils/net/iproute2/

21. Офіціальна сторінка kdb [Електронний ресурс]. – Режим доступу: http://ftp.altlinux.org/pub/people/legion/kbd

22. Less is a free, open-source file pager [Електронний ресурс]. – Режим доступу: http://www.greenwoodsoftware.com/less/

23. THE HOME OF LIBCAP [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/fullycapable/

24. A Portable Foreign Function Interface Library [Електронний ресурс]. – Режим доступу:https://sourceware.org/libffi/

25. Pipeline manipulation library [Електронний ресурс]. – Режим доступу:http://libpipeline.nongnu.org/

26. Офіціальна сторінка The Linux Kernel Archives [Електронний ресурс]. – Режим доступу:https://www.kernel.org/

27. Man-db, the on-line manual database [Електронний ресурс]. – Режим доступу:https://www.nongnu.org/man-db/

28. The Linux man-pages project [Електронний ресурс]. – Режим доступу:https://www.kernel.org/doc/man-pages/

29. The Meson Build system [Електронний ресурс]. – Режим доступу:https://mesonbuild.com

30. The GNU MPFR Library [Електронний ресурс]. – Режим доступу:https://www.mpfr.org/

31. Офіціальна сторінка ncurses [Електронний ресурс]. – Режим доступу:http://www.gnu.org/software/ncurses/

32. Офіціальна сторінка ninja [Електронний ресурс]. – Режим доступу: https://ninja-build.org/

33. Офіціальна сторінка openssl [Електронний ресурс]. – Режим доступу: https://www.openssl.org/

34. GNU patch [Електронний ресурс]. – Режим доступу: https://savannah.gnu.org/projects/patch/

35. Офіціальна сторінка Perl [Електронний ресурс]. – Режим доступу: https://www.perl.org/

36. Офіціальна сторінка pkg-conf [Електронний ресурс]. – Режим доступу: https://www.freedesktop.org/wiki/Software/pkg-config

37. Архів procps-ng [Електронний ресурс]. – Режим доступу: https://sourceforge.net/projects/procps-ng

38. Git psmis [Електронний ресурс]. – Режим доступу: http://psmisc.sourceforge.net/

39. Офіціальна сторінка Python [Електронний ресурс]. – Режим доступу: https://www.python.org/

40. The GNU Readline Library [Електронний ресурс]. – Режим доступу: https://tiswww.case.edu/php/chet/readline/rltop.html

41. GNU sed [Електронний ресурс]. – Режим доступу: http://www.gnu.org/software/sed/

42. Systemd System and Service Manager [Електронний ресурс]. – Режим доступу: https://www.freedesktop.org/wiki/Software/systemd/

43. The source code release home for Tcl (Tool Command Language) and the Tk toolkit [Електронний ресурс]. – Режим доступу: http://tcl.sourceforge.net/

44. Time Zone Database [Електронний ресурс]. – Режим доступу: https://www.iana.org/time-zones

45. Офіціальна сторінка util-linux [Електронний ресурс]. – Режим доступу: http://freecode.com/projects/util-linux

46. Офіціальна сторінка [Електронний ресурс]. – Режим доступу: https://www.vim.org

47. GitHub XML-Parser [Електронний ресурс]. – Режим доступу: https://github.com/chorny/XML-Parser

48. Офіціальна сторінка XZ Utils [Електронний ресурс]. – Режим доступу: https://tukaani.org/xz

49. Офіціальна сторінка zlib [Електронний ресурс]. – Режим доступу: https://www.zlib.net/

50. Debian Wiki Create Local Repo [Електронний ресурс]. – Режим доступу: https://wiki.debian.org/ru/CreateLocalRepo#A.2BBB8EPgQ0BD8EOARBBEw_.2BBDsEPgQ6BDAEOwRMBD0EPgQzBD4_.2BBEAENQQ.2FBD4ENwQ4BEIEPgRABDgETw_.2BBEEEMgQ.2BBDgEPA_.2BBDoEOwROBEcEPgQ8-

51. Debian Wiki налаштування дзеркала архіву Debian [Електронний ресурс]. – Режим доступу: https://www.debian.org/mirror/ftpmirror

52. Сторінки керівництва - DEBMIRROR [Електронний ресурс]. – Режим доступу: http://manpages.ylsoftware.com/ru/debmirror.1.html