МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

**С.Б. Могильний**

# ІНФОРМАЦІЙНА БЕЗПЕКА
## ЛАБОРАТОРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського*
*як навчальний посібник для здобувачів ступеня бакалавра за освітніми програмами*
*«Радіотехнічні комп'ютеризовані системи», «Інформаційна та комунікаційна*
*радіоінженерія», «Інтелектуальні технології радіоелектронної техніки»*
*спеціальності 172 «Електронні комунікації та радіотехніка»*

Київ
КПІ ім. Ігоря Сікорського
2023

Рецензент — *Мосійчук Віталій Сергійович*, канд. техн. наук, доц. кафедри прикладної радіоелектроніки радіотехнічного факультету, Національний технічний університет КПІ ім. Ігоря Сікорського

Відповідальний редактор — *Жук Сергій Якович*, д-р техн. наук, проф.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 9 від 22.06.2023 р.) за поданням Вченої ради радіотехнічного факультету (протокол № 06 /2023 від 29.05.2023 р.)*

Електронне мережне навчальне видання

Автор: *Могильний Сергій Борисович*, канд. техн. наук, доц.

# ІНФОРМАЦІЙНА БЕЗПЕКА
## ЛАБОРАТОРНИЙ ПРАКТИКУМ

Інформаційна безпека: Лабораторний практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 172 «Електронні комунікації та радіотехніка» / КПІ ім. Ігоря Сікорського; автор: С.Б.Могильний. – Електронні текстові дані (1 файл: 5,1 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 145 с.

В навчальному посібнику наводяться рекомендації до виконання лабораторних робіт з кредитного модуля «Інформаційна безпека», який викладається студентам радіотехнічного факультету, що навчаються на спеціальності 172 «Електронні комунікації та радіотехніка».

Реєстр. № НП 22/23-890. Обсяг 5,2 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

| | |
|---|---|
| ЛР | лабораторна робота |
| ПЗ | програмне забезпечення |
| AES | Advanced Encryption Standard |
| AP | Access Point |
| BSSID | Basic Service Set Identifier |
| CIS | COMODO Internet Security |
| DAQ | Data AcQuisition |
| DH | Diffie-Hellman |
| DNS | Domain Name System |
| EFS | Encrypted File System |
| FVEK | Full Volume Encryption Key |
| IDS | Intrusion Detection System |
| MOTD | Message Of The Day |
| Nmap | Network Mapper |
| PTK | Pairwise Transient Key |
| RPi | Raspberry Pi |
| SRK | Storage Root Key |
| SSID | Service Set Identifier |
| SSL | Secure Sockets Layer |
| SSH | Secure Shell |
| TPM | Trusted Platform Module |
| VMK | Volume Master Key |

# ВСТУП

Даний навчальний посібник призначений для підготовки бакалаврів за освітніми програмами «Радіотехнічні комп'ютеризовані системи», «Інформаційна та комунікаційна радіоінженерія», «Інтелектуальні технології радіоелектронної техніки» спеціальності 172 «Електронні комунікації та радіотехніка» і може бути використаний для інших освітніх програм.

Підготовка спеціалістів за дисципліною «Інформаційна безпека» передбачає 18 годин лекцій, 36 годин лабораторних занять, модульну контрольну роботу та РГР, які орієнтовані на отримання знань і навичок роботи з програмним забезпеченням для моніторингу, тестування та захисту комп'ютерних мереж і бездротових каналів передачі даних. Розглядаються методи побудови системи захисту від проникнення, використання шифрування для захисту документів на локальному комп'ютері.

Основна мета посібника – сформувати у студентів в процесі виконання лабораторних робіт навички реалізації на мікрокомп'ютері систем захисту від проникнення, VPN, інструментів пентестера тощо. Такі задачі вимагають знань системного проєктування, програмування, протоколів обміну даними, алгоритмів роботи різних виконавчих механізмів. Виконання лабораторних робіт дозволяє закріпити теоретичні знання, отримані на лекціях.

Навчальний посібник підготовлено відповідно до робочої навчальної програми (силабусу) дисципліни «Інформаційна безпека». Він містить необхідний теоретичний матеріал, який дозволяє застосовувати набуті знання для реалізації взаємодії мікрокомп'ютера з різними складовими систем захисту інформації.

Змістом даного видання є методичні рекомендації до виконання та опис лабораторних робіт кредитного модуля «Інформаційна безпека». У вказівках до кожної лабораторної роботи наведені теоретичні питання в необхідному для виконання лабораторних робіт об'ємі.

## Виконання лабораторних робіт

Перед виконанням лабораторних робіт студенти повинні самостійно вивчити відповідні розділи дисципліни за рекомендованою літературою і конспектом лекцій, зміст лабораторної роботи, порядок її виконання. В режимі ofline перед початком виконання кожної роботи студенти проходять контроль знань, на якому повинні показати розуміння мети і змісту роботи. Студенти, які отримали незадовільну оцінку, до лабораторної роботи не допускаються. Приступати до виконання лабораторної роботи можна тільки з дозволу викладача. Результати виконання лабораторної роботи у вигляді працюючих схем, часових діаграм при симуляції реалізованого проєкта тощо надаються викладачу.

## Зміст та оформлення звіту

У звіті необхідно відобразити:

- мету роботи;
- стислі теоретичні відомості;
- алгоритм та коди сценаріїв;
- аналіз отриманих результатів та висновки.

Звіти з лабораторних робіт виконуються кожним студентом індивідуально на окремих аркушах формату А4.

На координатних осях графіків повинні бути позначенням масштабу та розмірності вимірюваних величин.

У формулах та електричних схемах необхідно використовувати умовні позначення, які відповідають стандартам ЄСКД або використаної САПР. Особливу увагу необхідно приділити аналізу отриманих результатів та формулюванню висновків за результатами роботи.

Титульний лист звіту оформляється наступним чином:

Національній технічних університет України
Київський політехнічних інститут імені Ігоря Сікорського
Радіотехний факультет
Кафедра радіотехнічних систем

ЗВІТ

про лабораторну роботу ___
з дисципліни «Інформаційна безпека»

_____

(назва роботи)

студента ____ курсу, групи_____

_____

(прізвище, ім’я та по-батькові)

Викладач Могильний С.Б.:

Дата_____

202_ р.

Для забезпечення дистанційного (online) навчання використовуються платформи – Zoom та Moodle, які дозволяють проводити дистанційно лекційні та вступні заняття перед роботою (Zoom). Також при дистанційному навчанні студенти отримують цілодобовий доступ до мікрокомп’ютерів з Інтернету. Роз’яснення (консультації) зі складних та незрозумілих питань проводяться на онлайн зустрічах в Zoom.

Контроль виконання робіт та їх захист здійснюється на платформі дистанційного навчання Moodle. Протоколи виконання робіт завантажуються у відповідний розділ дисципліни в Moodle. Треба звернути увагу при оформлені

звітів робіт на дотримання вимог ДСТУ 3008:2015. Посилання на дистанційний курс – http://iot.kpi.ua/lms/course/view.php?id=2.

Зв'язок зі студентами підтримується через Телеграм–групу «Інформаційна безпека» та електронну пошту старост груп, що забезпечує своєчасну видачу та контроль індивідуальних завдань у відповідності до графіку навчання.

Отримані бали відповідно до рейтингової системи оцінювання накопичуються в журналі оцінок Moodle та в розділі «Поточна інформація» Кампуса КПІ (https://ecampus.kpi.ua). В Кампусі також викладені всі навчальні матеріали курсу.

## Лабораторна робота 1

# ВИКОРИСТАННЯ ВБУДОВАНИХ В ОС WINDOWS МЕТОДІВ ШИФРУВАННЯ

**Мета роботи:** Ознайомитися з методами та налаштуванням програмного забезпеченням Microsoft для шифрування, щоб захистити дані на локальному комп'ютері.

**Зміст.** В даній роботі вивчаються можливості Microsoft Office, EFS та BitLocker для захисту даних від несанкціонованого доступу до них.

### 1.1. Використання Microsoft Office

Шифрування папок і окремих файлів є надійним гарантом їх безпеки. Існує кілька методів, які користувачі Windows можуть використати для захисту своїх пристроїв і даних, які зберігаються на них.

Найбільш поширеними даними, які потрібно шифрувати, є документи Office. На щастя, Microsoft Office має свій власний інструмент шифрування. Для його використання потрібно:

- Відкрити документ за допомогою відповідної програми Office.
- Перейти на вкладку «Файл», потім вибрати розділ «Відомості» і пункт «Захист документа».
- У випадаючому меню натиснути на рядок «Зашифрувати за допомогою пароля».
- Ввести пароль, натиснути «Enter», а потім вказати пароль повторно.
- Натиснути «Ок».

Тепер документ Office зашифрований. Для отримання до нього доступу необхідний пароль, призначений для шифрування.

### 1.2. Система шифрування EFS

Це найшвидший спосіб шифрування в Windows. EFS є вбудованим інструментом, який використовує файлова система NTFS. Замість шифрування

всього диска, EFS дозволяє захищати каталоги та окремі файли. В процесі роботи Windows створює ключ шифрування, який зберігається локально. Процес простий, але не гарантує повну безпеку.

Роботу з EFS треба починати через вкладку «Властивості» файла/папки. Вибравши у «Властивостях» розділ «Атрибути стиснення і шифрування», треба поставити прапорець біля пункту «Шифрувати зміст для захисту даних». Після цього натиснути «ОК» і закрити вікно властивостей. Тепер назва зашифрованої папки відображається зеленим кольором.

**Увага:** перш ніж використовувати EFS, треба пам’ятати, що доступ до зашифрованої інформації можливий лише з паролем і логіном облікового запису. Варто записати ці дані, оскільки у випадку їх втрати інформація залишиться заблокованою назавжди.

**Використання EFS в Windows 10**

Опція EFS не доступна в домашніх версіях Windows 10, 8 або 7. Саме шифрування втрачає сенс, якщо ми не будемо використовувати надійний пароль зі своїм обліковим записом, тому що доступ до зашифрованих даних відкривається автоматично при вході в систему користувача, який їх зашифрував. В EFS використовується один алгоритм, це DESX – спеціальна модифікація розповсюдженого стандарту DES.

Після шифрування файлів система автоматично створить резервний ключ, який дозволить отримати доступ до даних, якщо у нас виникнуть проблеми з входом в систему, і ми не зможемо увійти в свій обліковий запис. Для цього треба мати будь-який зовнішній носій, краще всього використати USB флешку.

**Як включити EFS в Windows 10**

Виберемо файли або папки для шифрування з EFS та виконаємо дії:

1. Запустимо Windows провідник в меню **«Пуск»** на рабочому столі або на панелі задач.

2. Клацнемо правою кнопкою миші файл або папку і виберемо **«Властивості»**.

3. На вкладці Загальні натиснемо кнопку **«Інші»**.

4. Встановимо прапорець для **Шифрувати вміст для захисту даних**.

5. Натиснемо **«ОК»**, потім **«Застосувати»**. З'явиться вікно з питанням, чи хочемо ми зашифрувати лише вибрану папку чи папку і вкладені в неї інші папки та файли.

6. Натиснемо **«Застосувати зміни лише до цієї папки»** або **«Застосувати зміни до цієї папки, підпапок і файлів»**.

7. Натиснути **«ОК»**.

Файли, які ми зашифрували за допомого EFS, в правому верхньому куті іконки будуть мати значок маленького замка. Після цього нікто не зможе отримати до них доступ, поки не ввійде в систему під нашим обліковим записом.

**Створення резервної копії ключа шифрування EFS**

Після вмикання EFS на панелі задач в правому нижньому куті екрана з'явиться невеликий значок. Це нагадування нам про необхідність резервного копіювання ключа шифрування EFS.

1. Підключимо USB-накопичувач до комп'ютера.

2. Натиснемо значок EFS на панелі задач.

3. Натиснемо **«Архівувати зараз (рекомендується)».**

4. Натиснемо **Далі**. Відкриється вікно майстра експорту сертифікатів.

5. Натиснемо **Далі**. Відкриється вікно з встановленням формату файла.

6. Встановимо прапорець поруч з паролем, потім введемо і підтвердимо пароль. Це буде пароль для захисту архіва з копією ключа шифрування. Натиснути **Далі** для продовження.

7. Натиснемо **Огляд**, перейдемо на USB-накопичувач і збережемо файл.

Якщо ми втратимо доступ до свого облікового запису користувача, для доступу до зашифрованих файлів можна буде використати резервний ключ.

## 1.3. BitLocker для шифрування дисків

BitLocker – це система повного шифрування диску. Програма призначена для захисту даних методом шифрування цілих томів. Щоб розблокувати диск, захищений за допомогою BitLocker, треба ввести пароль або використати зовнішній USB-диск.

BitLocker може використовувати довірений платформенний модуль – мікросхему TPM, який дозволяє комп’ютеру підтримувати розширені функції безпеки. Але, оскільки не на всіх материнських платах присутній TPM, можливе використання BitLocker без цієї мікросхеми.

**Налаштування та робота з BitLocker**

BitLocker дозволяє захищати дані шляхом повного шифрування диска (логічних, починаючи з Windows 7, карт SD та USB-флешок). Підтримуються наступні алгоритми шифрування:

- AES 128
- AES 128 с Elephant diffuser (використовується за замовчуванням)
- AES 256
- AES 256 с Elephant diffuser

Сам ключ може зберігатися в TPM або на USB-пристрої, чи на комп'ютері. У випадку з TPM при завантаженні комп'ютера ключ може бути отриманий з нього відразу, або тільки після аутентифікації за допомогою USB-ключа чи введення PIN-коду користувачем. Таким чином, можливі наступні комбінації, щоб отримати доступ:

- TPM
- TPM + PIN
- TPM + PIN + USB-ключ
- TPM + USB-ключ
- USB-ключ (даний режим вимагає активації через групові політики)
- Пароль (даний режим доступний, починаючи з Windows 8, а також вимагає активації через групові політики)

**Принципи роботи BitLocker**

BitLocker шифрує том, а не фізичний диск. Том може займати частину диска, а може включати в себе масив з декількох дисків. Для роботи BitLocker в разі шифрування системного диска буде потрібно два NTFS-томи, один для ОС і один для завантажувальної частини. Останній повинен бути не менше 1,5 Гб і не бути зашифрований.

Починаючи з Windows Vista SP1 з'явилася можливість шифрувати несистемні томи. Після створення розділів необхідно ініціалізувати TPM-модуль в ПК, де він є, і активувати BitLocker. У Windows 7 з'явився BitLocker To Go, що дозволяє шифрувати змінні носії, а також знижені вимоги для завантажувальної частини, для неї достатньо 100 Мб.

Ця технологія використовує алгоритм AES (Advanced Encryption Standard). Ключі шифрування повинні зберігатися безпечно і для цього в BitLocker є кілька механізмів. Найпростіший, але водночас і найменш безпечний метод – це пароль. Ключ генерується з пароля щоразу однаковим чином і, відповідно, якщо хтось дізнається ваш пароль, то і ключ шифрування стане відомий.

Щоб не зберігати ключ у відкритому вигляді, його можна шифрувати або в TPM (Trusted Platform Module), або на криптографічному токені чи смарт-карті, що підтримує алгоритм RSA 2048. Використання смарт-карти або токена для зняття блокування диска є одним з найбезпечніших способів, що дозволяють контролювати, хто виконав цей процес і коли. Для зняття блокування в такому випадку потрібно як сама смарт-карта, так і PIN-код до неї.

Схема роботи BitLocker (рис.1.1):

1. При активації BitLocker за допомогою генератора псевдовипадкових чисел створюється головна бітова послідовність. Це ключ шифрування тома – FVEK. Їм шифрується вміст кожного сектора. Ключ FVEK зберігається в суворій секретності.

2. FVEK шифрується за допомогою ключа VMK. Ключ FVEK (зашифрований ключем VMK) зберігається на диску серед метаданих тома. При цьому він ніколи не повинен потрапляти на диск в розшифрованому вигляді.

3. Сам VMK теж шифрується. Спосіб його шифрування вибирає користувач.

4. Ключ VMK за замовчуванням шифрується за допомогою ключа SRK, який зберігається на криптографічній смарт-карті або токені. Аналогічним чином це відбувається і з TPM.

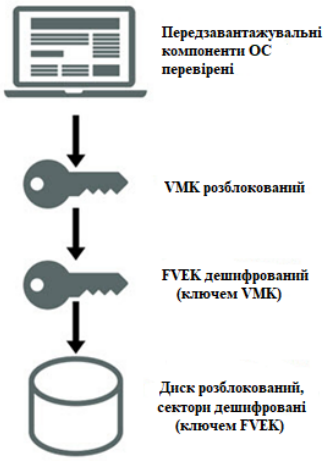


Рисунок 1.1 – Послідовність доступу до зашифрованих даних

До слова, ключ шифрування системного диска в BitLocker можна захистити за допомогою смарт-карти або токена. Це пов'язано з тим, що для доступу до смарт-картки і токена використовуються бібліотеки від вендора, і до завантаження ОС, вони, ясна річ, не доступні. Якщо немає TPM, то BitLocker пропонує зберегти ключ системного розділу на USB-флешці, а це, звичайно, не найкраща ідея. Якщо у нашій системі немає TPM, то не рекомендується шифрувати системні диски.

І взагалі, шифрування системного диска при цьому є поганою ідеєю. При правильному налаштуванні всі важливі дані зберігаються окремо від системних. Це як мінімум зручніше з точки зору їх резервного копіювання. Плюс шифрування системних файлів знижує продуктивність системи в цілому, а робота незашифрованого системного диска з зашифрованими файлами відбувається без втрати швидкості.

5. Ключі шифрування інших несистемних і знімних носії можна захистити за допомогою смарт-карти або токена, а також TPM. Якщо ні модуля TPM, ні смарт-карти немає, то замість SRK для шифрування ключа VMK

використовується ключ згенерований на основі введеного нами пароля. При запуску з зашифрованого завантажувального диска система опитує всі можливі сховища ключів: перевіряє наявність TPM, перевіряє USB-порти або, якщо необхідно, запитує користувача (що називається відновленням). Виявлення сховища ключа дозволяє Windows розшифрувати ключ VMK, яким розшифровується ключ FVEK, а далі вже ним розшифровуються дані на диску.

Кожен сектор тому шифрується окремо, при цьому частина ключа шифрування визначається номером цього сектора. В результаті два сектори, що містять однакові незашифровані дані, будуть в зашифрованому вигляді виглядати по-різному, що сильно ускладнить процес визначення ключів шифрування шляхом запису і розшифровки заздалегідь відомих даних.

Крім FVEK, VMK і SRK, в BitLocker використовується ще один тип ключів, створюваний «про всяк випадок». Це ключі відновлення. Для аварійних випадків (користувач втратив токен, забув його PIN-код тощо) BitLocker на останньому кроці пропонує створити ключ відновлення. Відмова від його створення в системі не передбачена.

**Механізми розшифровки та їх уразливості**

Існує три механізми перевірки автентичності, які можна використовувати для реалізації Bitlocker-шифрування:

- *Прозорий режим роботи.* Цей режим використовує можливості апаратного забезпечення Trusted Platform Module для надання прозорої роботи користувачів. Користувачі входять на комп'ютер з операційною системою Windows в звичайному режимі. Ключ, використовуваний для шифрування диска, закодований в чіп TPM і може бути виданий тільки в коді завантажувача ОС (якщо завантажувальні файли показуються як незмінені). Цей режим вразливий для нападу при холодному завантаженні, так як дозволяє зловмисникові вимкнути комп'ютер і завантажитися.
- *Режим перевірки автентичності користувача.* Цей режим передбачає, що користувач пройшов деяку аутентифікацію в

передзавантажувальному середовищі у вигляді попереднього введення PIN-коду. Цей режим вразливий до буткіт-атак.

- *Режим USB-ключа.* Для можливості завантаження в захищену операційну систему користувач повинен вставити в комп'ютер пристрій USB, який містить ключ запуску. Звернемо увагу, що для цього режиму необхідно, щоб BIOS на комп'ютері підтримував читання пристроїв USB в завантажувальному середовищі. Цей режим також вразливий до буткіт-атак.

**Використання в інших операційних системах**

Існує неофіційна утиліта dislocker для операційних систем GNU/Linux і macOS, яка представляє собою інструмент для читання і запису томів, зашифрованих через BitLocker. Для надання доступу до зашифрованого тому утиліта dislocker використовує FUSE-драйвер, а створювати нові зашифровані томи дана утиліта не вміє.

Основна відмінність для користувача (або системного адміністратора) від відомої функції шифрування, вбудовану в файлову систему NTFS засіб EFS – те, що в EFS необхідно вручну вказувати, які файли треба шифрувати, BitLocker же здійснює шифрування «прозоро», відразу при записі, при мінімумі втручань з боку користувача. BitLocker не шифрує наступні елементи: метадані, завантажувальні сектора та пошкоджені сектора.

Слід мати на увазі, що BitLocker прив'язується до певного ПК, і на іншому ПК (якщо він тільки не буде 100% ідентичний нашому, що нереально), дані буде неможливо (або надзвичайно важко) прочитати навіть з використанням USB ключа.

Алгоритм, який застосовується для шифрування – AES з довжиною ключа 128 або 256 біт, можна змінити за допомогою групових політик або WMI. У таблиці 1 наведені основні переваги і недоліки системи BitLocker.

Таблиця 1.1 – Переваги та недоліки системи шифрування BitLocker

| Переваги | Недоліки |
|---|---|
| Можливість контролю через Active Directory. | Слабка попередня аутентифікація – можливе створення руткіта, що змінює завантажувальні файли ОС – буде потрібно відновлення даних (при використанні без мікросхеми TPM). |
| Робота алгоритму на рівні томів, що забезпечує високу ступінь стійкості. | Досить середня підтримка багатофакторної аутентифікації (тільки TPM і USB-ключ) |
| Можливість використання апаратного засобу захисту – мікросхеми TPM. | Складнощі в налаштуванні. |
| Висока надійність. Заснований на алгоритмі AES, що вважається алгоритмом з високим показником стійкості до злому. | Тісно інтегрований в ОС, що може призвести до витоку даних при зломі. |
| Комп'ютер з мікросхемою TPM може перевіряти цілісність завантажувальних компонентів до запуску ОС. | Потрібно сумісний за ОС комп'ютер. |
| Ідеальне рішення для захисту від Offline атак (атаки з вимиканням).<br>Вартість системи включена у вартість ОС<br>Висока продуктивність, непомітний при роботі. | В нових системах можуть проявитися помилки розробки. |

**Шифрування диска в Windows 10 з модулем TPM**

Якщо на материнській платі присутній модуль TPM, шифрування жорсткого диску в Windows 10 організувати дуже просто:

1. Натиснемо кнопку **Пуск**, відкриємо **Провідник** і виберемо **Цей комп'ютер**.

2. У вікні клацнемо правою кнопкой миші на диску, який хочемо зашифрувати, і у випадаючому меню виберемо **Увімкнути BitLocker**.

3. Введемо надійний пароль до диску. Щоразу, коли будемо вмикати комп'ютер, Windows буде запитувати цей пароль, щоб розшифрувати дані.

4. Виберемо, яким чином треба створити резервну копію ключа відновлення. Його можна зберегти в обліковому запису Microsoft, копіювати на USB-накопичувач або роздрукувати.

5. Виберемо, яку частину диску шифрувати: весь чи лише вільне місце. Якщо ми недавно встановили Windows 10, обираємо друге. Якщо ж вмикаємо шифрування на диску, який вже використовується, краще зашифрувати весь диск.

6. Натиснемо **Продовжити**, щоб почати шифрування.

7. Коли шифрування завершиться, перезавантажимо комп'ютер і введемо пароль.

Якщо на кроці 2 з'явиться повідомлення про помилку, яка вказує, що нам треба дозволити запуск BitLocker без сумісного модуля TPM, це означає, що наша материнська плата не має відповідного модуля. В цьому випадку доведеться використати інший спосіб.

**Шифрування диска в Windows 10 без модуля TPM**

Для того, щоб зашифрувати жорсткий диск в Windows 10 без допомоги апаратного модуля зберігання ключів, зробимо наступне (числа на рисунках показують послідовність наших дій):

1. З меню **Система Windows** > **Командний рядок** > **Більше** > **У режимі адміністратора** (рис.1.2) відкриваємо вікно із командним рядком, запущеним у режимі адміністратора.

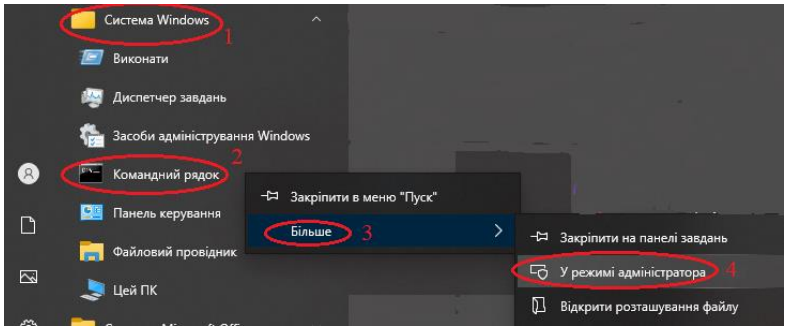


Рисунок 1.2 – Запуск командного рядка

2. У командному рядку вводимо **mmc** (рис.1.3) і відкриваємо вікно Microsoft Management Console, призначеної для створення, збереження та запуску адміністративних інструментів.

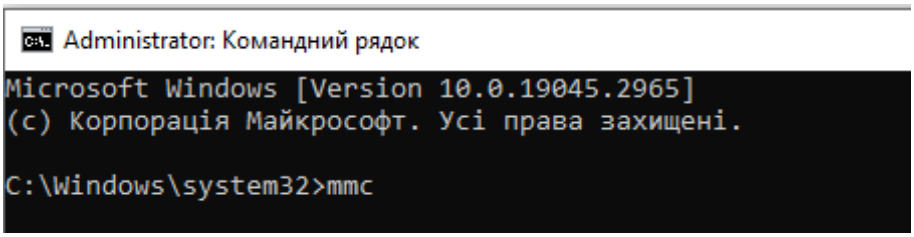


Рисунок 1.3 – Запуск Microsoft Management Console

3. З меню консолі **Файл** > **Додати/видалити оснастку…** (рис. 1.4) відкриваємо вікно для роботи з оснастками.

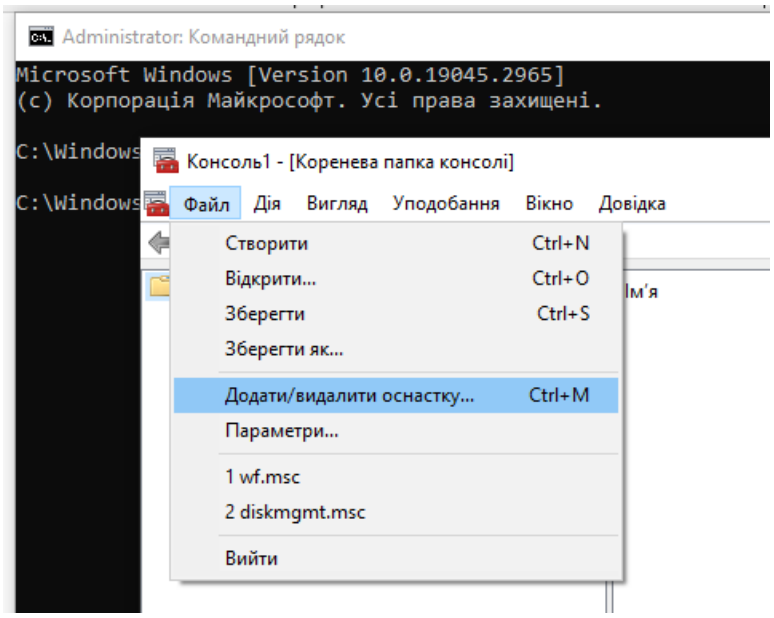


Рисунок 1.4 – Меню додавання/видалення оснастки

4. У вікні роботи з оснастками (рис.1.5) знаходимо редактор об’єктів групової політики і додаємо його для об’єкта **Локальний комп’ютер**.

5. Перейдемо в **Конфігурація комп’ютера** >**Адміністративні шаблони** > **Компоненти Windows** > **Шифрування диска BitLocker** > **Диски операційної системи** (рис.1.6).

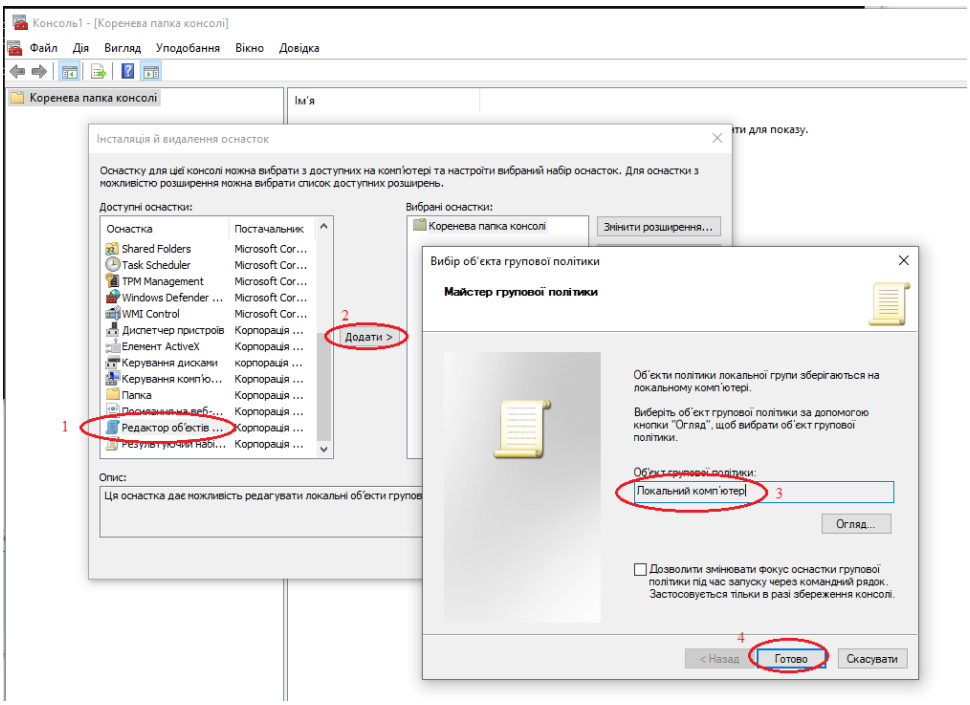


Рисунок 1.5 – Додавання редактора об’єктів групової політики

6. Двічі клацнемо на **Вимагає додаткової перевірки справжності при запуску** (рис.1.6).

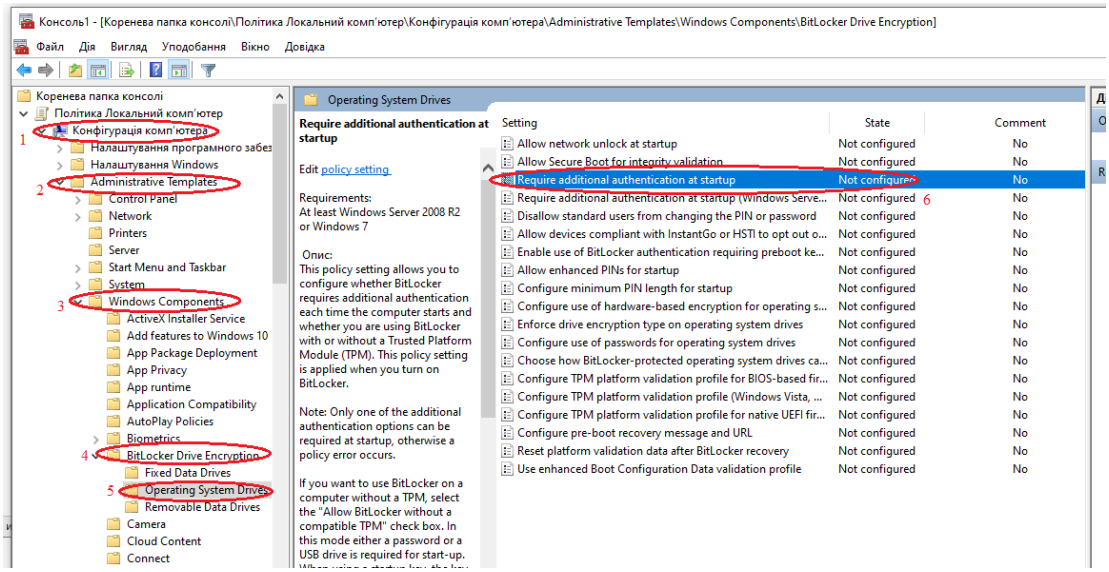


Рисунок 1.6 – Вибір параметрів шифрування для BitLocker

7. В новому вікні виберемо пункт **Дозволено**, поставимо галочку напроти **Дозволити використання BitLocker без сумісного довіренного платформенного модуля** і натиснемо **ОК**.

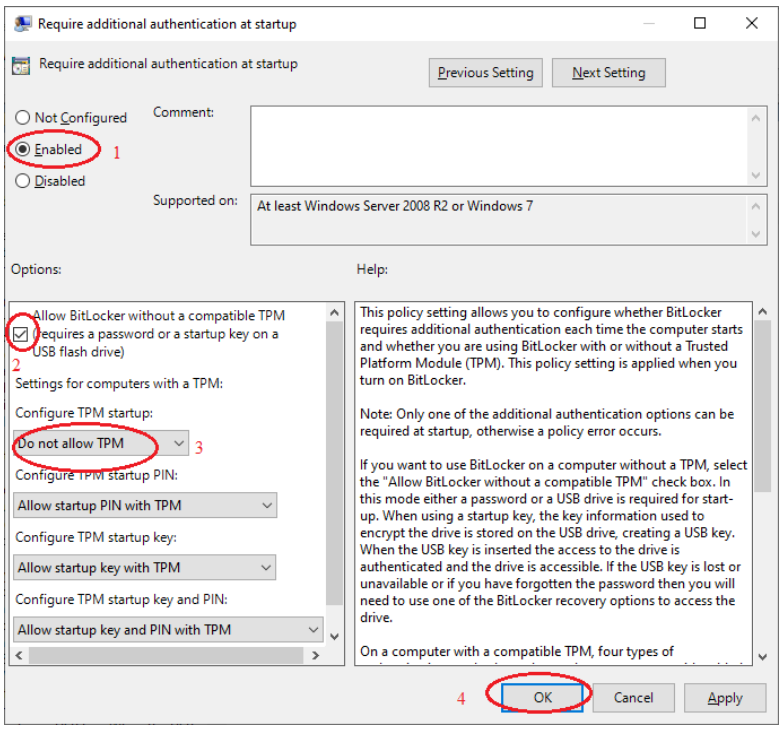


Рисунок 1.7 – Вибір дозволу BitLocker без модуля TPM

8. Відкриємо **Цей комп'ютер**, виберемо диск і натиснемо **Увімкнути BitLocker**, як описано в першому способі.

Після цього на нашому комп'ютері буде виконана швидка перевірка. Коли перевірка закінчиться, система запитає, чи хочемо заблокувати свій комп'ютер за допомогою USB-ключа або пароля.

Виберемо, чи буде BitLocker шифрували вільне місце, що залишилось, чи весь жорсткий диск.

BitLocker запуститься у фоновому режимі, виконуючи шифрування нашого жорсткого диску. Ми можемо продовжити працювати у нормальному режимі.

Після першого перезавантаження ПК наша система буде завантажуватися, лише якщо ми введемо правильний пароль під час запуску або підключимо зовнішній носій з резервною копією ключа.

BitLocker і EFS – вбудовані компоненти Windows. Якою з них скористатися? BitLocker вважають більш безпечний. Але це залежить від особистих вподобань. Наприклад, для домашніх користувачів EFS цілком може бути достатньо.

**Короткі висновки**

1. Зашифровані файли не захищені на 100%. Злочинці можуть, хоча це й не просто, обійти такий захист. Збереження криптографічних ключів і паролів в незашифрованому вигляді, а також встановлені без відома користувача за допомогою шкідливого ПЗ кейлогери (keyloggers), несуть загрозу комп'ютерній безпеці. Наприклад, при використанні EFS комп'ютер зберігає незашифровану версію в тимчасовій пам'яті, тому злочинець може отримати до неї доступ. Якщо дані дійсно важливі, використовуємо платне, надійне рішення за допомогою хмарного шифрування.

2. Пам'ятаємо, що EFS файл втрачає шифрування після його копіювання на іншу файлову систему (FAT 32 або exFAT), а також при передачі через мережу або електронною поштою тим користувачем, який шифрував. Інші користувачі не зможуть копіювати зашифрований файл на інші файлові системи чи передати їх мережею.

3. Завжди створюємо незашифровані резервні копії для випадку втрати паролів. Зберігаємо їх в безпечному фізичному місці, бажано в автономному режимі. До речі, багато хто не використовує шифрування саме із-за страху втратити ключ до них.

4. Визначаємось, що саме хочемо/повинні шифрувати. Це визначить, який метод захисту буде використовуватися.

## 1.4. Завдання

1. Виконайте шифрування за допомогою BitLocker і EFS. Зробіть відповідні скріншоти для протоколу.

2. Знайдіть через редактор групової політики збережений сертифікат шифрування та зробіть скріншот параметрів шифрування, наведених в цьому сертифікаті.

3. Створіть в ОС Windows агента відновлення зашифрованих файлів, переконайтеся, що новий доданий сертифікат для «відновлення» (recovery) файлів. Зробіть скріншоти.

4. Протокол завантажте у відповідну папку платформи Moodle.

### 1.5. Зміст звіту

1. Мета роботи.
2. Скріншоти результатів виконання завдання.
3. Відповіді на контрольні питання до роботи.
4. Короткі висновки.

### 1.6. Контрольні питання

1. Який стандарт шифрування використовується в EFS?
2. Який стандарти шифрування використовуються в BitLocker?
3. Які функції виконує модуль TPM?
4. Яким ключем шифрується ключ шифрування тома?
5. Де зберігається ключ FVEK?

## Лабораторна робота 2

# НАЛАШТУВАННЯ SSH ДЛЯ РОБОТИ НАВІДДАЛЕНОМУ КОМП'ЮТЕРІ

**Мета роботи:** Вивчити, як налаштувати SSH-ключ на нашому локальному комп'ютері та використати згенеровані пари ключів для підключення до віддаленого сервера.

**Зміст.** В роботі розглядається технологія генерування ключа SSH та його використання для безпечного обміну даними в мережі.

## 2.1. Теоретичні відомості

Метод SSH є зручним і надає більш захищений шлях для підключення до віддаленого сервера або комп'ютера на відміну від простого використання пароля.

Встановити з'єднання SSH можна з PuTTY – програмою, яка дозволяє працювати на віддаленому RPi та інших комп'ютерах з ОС Linux з комп'ютера Windows PC, що особливо важливо із-за великої кількість користувачів даної ОС.

### Підготовка та встановлення

Все, що нам потрібно, це підключити до RPi живлення та Інтернет.

Спочатку нам необхідно дізнатися IP-адресу **RPi,** бо вона зазвичай надається динамічно через DHCP. Звичайно, вона нам вже відома, якщо ми встановили статичну IP-адресу.

Однак, при динамічній адресації маршрутизатор наступного разу може призначити іншу IP-адресу. Це проблема, тому що для підключення через SSH, ми повинні знати IP-адресу RPi у нашій локальній мережі. Ми не зможемо побачити її, якщо не матимемо моніторингу приєднаних пристроїв. Необхідне сканування мережі, щоб знайти IP-адресу RPi.

Звичайно, ми можемо дізнатися IP-адресу RPi, увійшовши в маршрутизатор і побачивши, які пристрої підключені. Але є простіший спосіб:

за допомогою безкоштовного інструмента під назвою Advanced IP Scanner. Завантажити його можна з radmin.com.

Коли ми запустимо IP Scanner, то побачимо екран, схожий на показаний нижче (рис.2.1). Натискаємо кнопку **Сканувати** (Scan):

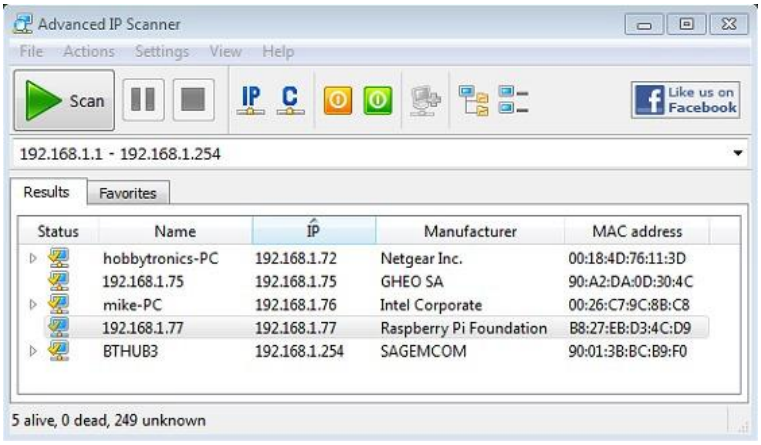


Рисунок 2.1 – Знаходження IP-адреси RPi

Серед відображених приєднаних до мережі пристроїв ми без проблем знайдемо свій RPi та його IP–адресу.

На наступному кроці переконаємося, що сервер SSH на RPi працює. Для цього введемо в терміналі сесії команду:

```
sudo service ssh status
```

Якщо сервер SSH запущений, то ми повинні отримати відповідь, що послуга `sshd` працює. Якщо це не так, то введемо в термінальному сеансі:

```
sudo raspi-config
```

Запуститься утиліта конфігурування ОС, яка входить в дистрибутив. Виберемо пункт меню для SSH, а потім виберемо дозвіл на вмикання сервера SSH. Перезавантажимо RPi.

Для використання PuTTY для завантаженння з сайту PuTTY доступні п'ять файлів:

- **PuTTY.exe** – клієнт Secure Shell.
- **PuTTYgen.exe** – генератор публічного/приватного ключів SSH.

• **Pagent.exe** – агент аутентифікації, зберігає ключі в пам’яті і при його використанні не треба щоразу вводити ключову парольну фразу.

• **PSCP.exe** – для захищеного копіювання з командного рядка.

• **PSFTP.exe** – для захищеного копіювання з FTP-подібним інтерфейсом.

Для уникнення проблем при встановленні, переконаємося, що каталог встановлення знаходиться в шляху для команди.

У Windows 10 клацнемо правою кнопкою миші на **Комп'ютер** в меню **Старт** і виберемо пункт **Властивості**. Внизу вікна натиснемо кнопку **Параметри середовища**. Відкриється діалогове вікно (рис.2.2):

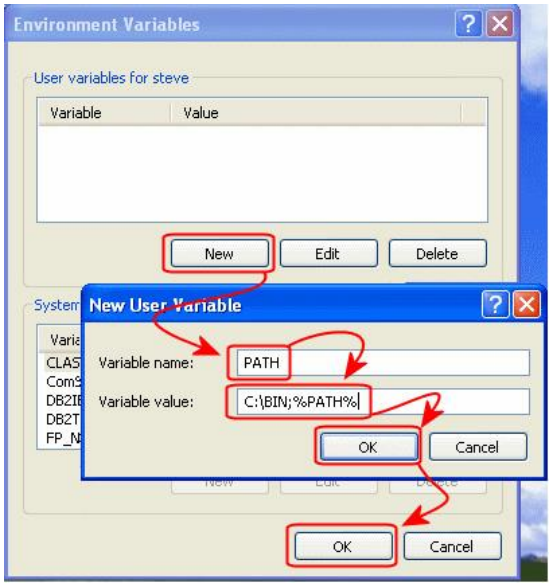


Рисунок 2.2 – Шлях до змінної Path

Завжди існує змінна **Path** в розділі **Системні змінні**, а іноді також і в розділі **Змінні середовища користувача**. Лише адміністратори мають доступ до системних змінних, щоб змінити або додати **Path**, при необхідності.

Ми поставимо нову папку типово на початку шляху і вона буде відокремлена від решти частини списку комою. Натиснемо **OK**, щоб зберегти всі зміни.

Встановлення PuTTY не має особливостей. Після закінчення встановлення створимо ярлик на робочому столі, бо ця програма часто використовується. В нашому випадку, шлях в ярлику до виконуваного файлу C:\Bin\putty.exe.

**Запуск PuTTY та налаштування для цільової системи**

Необхідні значення для початкового нлаштування:

**Сесія:** Ім'я вузла (Host Name): *ім'я вузла, наприклад,* dbserver (рис.2.3), або IP-адреса віддаленого комп'ютера.

**Протокол** (Protocol): SSH.

**З'єднання: Дані (Data).** Користувач з автологіном: *наприклад,* steve.

**З'єднання: SSH.** Віддати перевагу SSH протоколу версії: тільки 2.

**Вікно: Переклад (Translation).** За замовчуванням встановлено ISO-8859-1, тому повинні відкрити список і вибрати UTF-8.

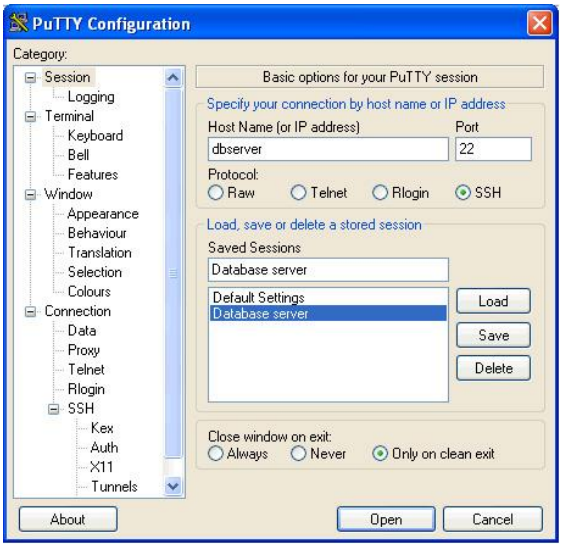


Рисунок 2.3 – Початкові налаштування

Вводимо відповідні дані, зберігаємо сесію (Save) і натискуємо **Open**, щоб відкрилося вікно авторизації (рис.2.4), і для входу використовуємо ім'я і пароль користувача (за замовчуванням для RPi: **pi** і **raspberry**).


Рисунок 2.4 – Вікно авторизації

При доступі до декількох віддалених систем можна створити і використовувати декілька сеансів, відповідним чином відредагувавши ярлики запуску збережених сеансів.

Для цього клацнемо правою кнопкою миші на ярлику і виберемо **Властивості** (рис.2.5), потім введемо параметр `-load` разом з ім'ям сесії (у лапках, якщо необхідно):

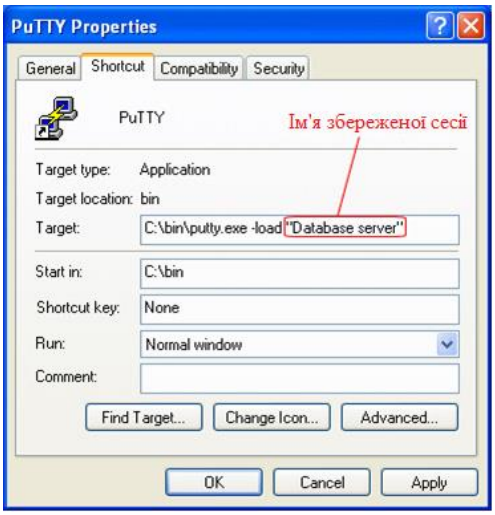


Рисунок 2.5 – Створення сеансу

Для роботи в термінальному режимі будуть корисними такі команди:

`man команда` – видає докладну допомогу по команді;

`ls` – список файлів;

`ls -lha` – показує всі файли (включаючи приховані), розмір файлів, права на них, дату останньої зміни;

`ls -lha | less` – дозволить переглядати файли посторінково (якщо їх багато);

`mv` – перейменувати, перенести;

`cp` – скопіювати;

`rm` – видалити;

`>` – очищення файлу. Цю команду можна застосовувати до лог-файлів. Якщо почистити файли з директорії `/site/наш_домен/log/` наступним чином:

```
> access.log
> error.log
> combined.log
```

то очистяться логи, а от їх видалення, як і видалення інших службових файлів, може призвести до припинення роботи нашого сайту.

`mc` – клон Нортон Командера, в якому зручно працювати з файлами, а також можна працювати з ними по FTP (FTP всередині SSH).

**Створення, встановлення та використання публічних/приватних ключів**

Для шифрування пропонуються на вибір два стандарти шифрування: DSA і RSA.

DSA або Digital Signature Algorithm – стандарт американського уряду, розроблений NSA для цифрових підписів. Він є похідним від алгоритму підпису Ель-Гамаля. Безпека будується на основі складності вирішення певних типів логарифмічних функцій.

RSA названий від ініціалів авторів: Ron Rivest, Adi Shamir і Leonard Adleman, які першими опублікували алгоритм. Він заснований на складності факторизації великих чисел.

Хоча DSA і RSA мають криптографію практично однакової потужності, але у кожного є свої переваги, коли беруть до уваги продуктивність. DSA швидше працює для розшифровки і підписання, а RSA – швидше при шифруванні та перевірці.

Тому на наш вибір, але рекомендується ключ від 2048 до 4096 біт (рис.2.6). Більш короткі ключі можуть бути недостатньо стійкими, а більш довгі дуже довго генеруються.

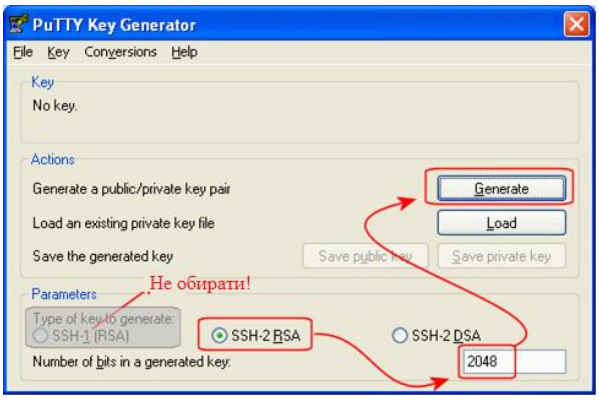


Рисунок 2.6 – Вибір стандарту шифрування

***Увага. Не створюємо ключі будь-якого типу SSH версії 1: вони не є безпечними!***

**Захист і збереження ключів**

Тепер пару ключів згенеровано, але вона існує лише в пам'яті PuTTYgen: її потрібно зберегти на диску, щоб бути корисною. Хоча відкритий ключ не містить конфіденційної інформації та буде встановлено на віддалених системах, приватний ключ необхідно надійно захищати: будь-хто, хто знає закритий ключ, має повний доступ до всіх віддалених систем. Приватний ключ зазвичай захищений парольною фразою, і ця фраза вводиться двічі у зазначені поля (рис.2.7). Коментар необов'язковий, але часто це адреса електронної пошти власника ключа. Це також може бути просто ім'я власника.

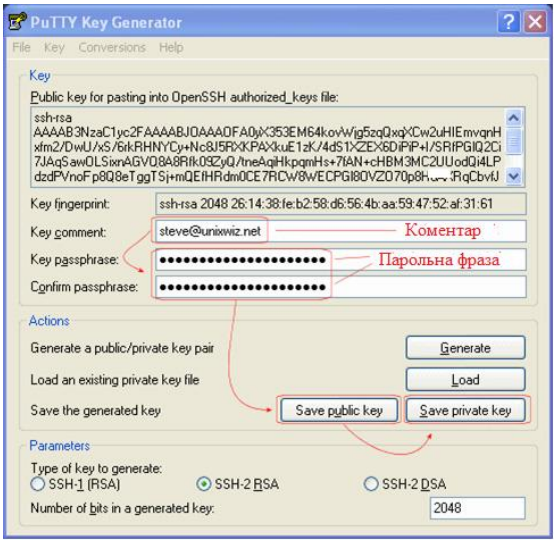


Рисунок 2.7 – Захист з парольною фразою

***Увага. Не забуваємо парольну фразу: пара ключів без неї марна!***

**Встановлення публічного ключа на RaspberryPi**

З ще відкритим PuTTYgen, виділяємо під заголовком «Публічний ключ для вставки в OpenSSH файл `authorized_keys`» всю площу (рис.2.8) і через Ctrl+C копіюємо в буфер локальної системи. По суті, це ті ж дані, які містяться в збереженому файлі приватного ключа, але це у формі, яка може бути безпосередньо використана на RPi.

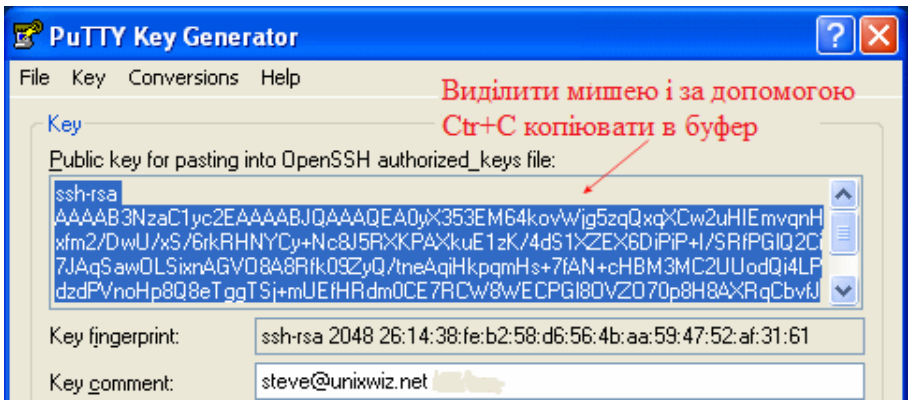


Рисунок 2.8 – Вікно копіювання публічного ключа

Входимо на RPi за допомогою облікового запису, створюємо каталог SSH, якщо необхідно:

```
sudo mkdir ~/.ssh
```

Редагуємо або створюємо файл з іменем `.ssh/authorized_keys`. Це буде текстовий файл і текст із буферу обміну повинен бути вставлений в нього. Для цього можна скористатися одним з 7 найбільш популярних способів:

1. `touch` *`ім'я_файла`* – створити пустий файл;
2. `cat >` *`ім'я_файла`* – і введення закінчити ctrl+z;
3. `vim` *`ім'я_файла`* – створити або редагувати *ім'я_файла;*
4. nano *ім'я_файла* – використання вбудовано редактора;
5. `echo "текст" >` *`ім'я_файла`*;
6. `>` *`ім'я_файла`* – мабуть, найбільш короткий спосіб;
7. `cp` *`ім'я_файла _0 ім'я_файла`* – скопіювати або створити файл.

Відкритий ключ має тільки один довгий рядок, і дуже легко вставити дані таким чином, що обрізаємо перші кілька символів. Це робить ключ непрацездатним, так що переконаємося, що ключ починається з SSH-RSA або DSA-SSH (рис.2.9). Збережемо файл.

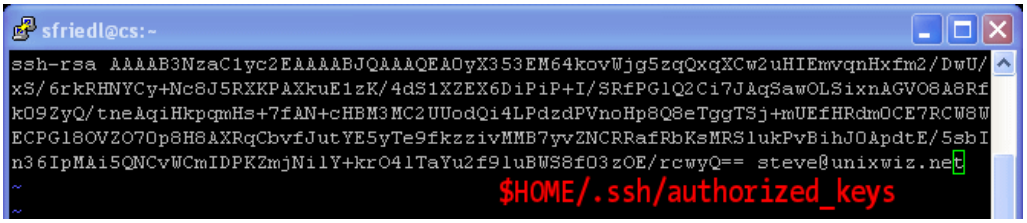


Рисунок 2.9 – Вікно вставлення публічного ключа

Також подбаємо про захист збереженого файла. Переконаймося, що каталог SSH і файли в ньому може прочитати тільки їх власник – це міра безпеки, яка може бути досягнута за допомогою параметрів команди `chmod` із застосуванням до всього каталогу:

```
sudo chmod 700 ~/.ssh
```

Права доступу записуються в `chmod` одним рядком одночасно для трьох типів користувачів:

- власника файла;
- інших користувачів, які входять в групу власника;
- всіх решти користувачів.

Аргумент команди `chmod`, який задає дозвіл, може бути записаний в двох форматах: в числовому або в символьному.

Для директорій зі всього списку зазвичай застосовують лише `0`, `5` і `7` – повна заборона, читання і повний доступ

В нашому випадку `700 (-rwx------)` означає, що власник може читати, записувати і запускати для виконання, а ніхто інший не має права виконувати будь-які дії. В табл.7.1 показаний приклад права `755`.

Таблиця 2.1 – Приклад значення прав `755`

| | Власник | Група | Решта |
|---|---|---|---|
| Восьмеричне значення | 7 | 5 | 5 |
| Символьний запис | rwx | r-x | r-x |
| Позначення типу користувача | u | g | o |

Виконуємо перевірку за допомогою:

```
ls-lR~/.ssh
```

Результати виконання команди:

```
/home/steve/.ssh:
total 16
drwx------    2 steve    steve        4096 Nov 22 13:11 ./
drwx------    6 steve    steve        4096 Nov 22 16:10 ../
```

```
-rw-------      1   steve      steve          1150   Nov   22   13:11
authorized_keys
```

Тепер можна вийти із системи.

**Прикріплення приватного ключа сесії SSH**

Тепер, коли публічні/приватні ключі створені, вони можуть бути пов'язані з сесією SSH. По-перше, ми зробимо це в PuTTY, запустивши програму та завантаживши необхідну сесію.

Перейдемо до **З'єднання** (Connection): **SSH**: панель **Auth** в категорії ліворуч (рис.2.10), потім заповнимо файл приватного ключа для поля аутентифікації, перейшовши до файлу **.ppk**, збереженого раніше. Повернемося до рівня категорії сесії і збережемо поточну сесію.

На даний момент, клієнт PuTTY (на Windows) і сервер OpenSSH (на RPi) налаштовані на безпечний доступ з публічним ключем.

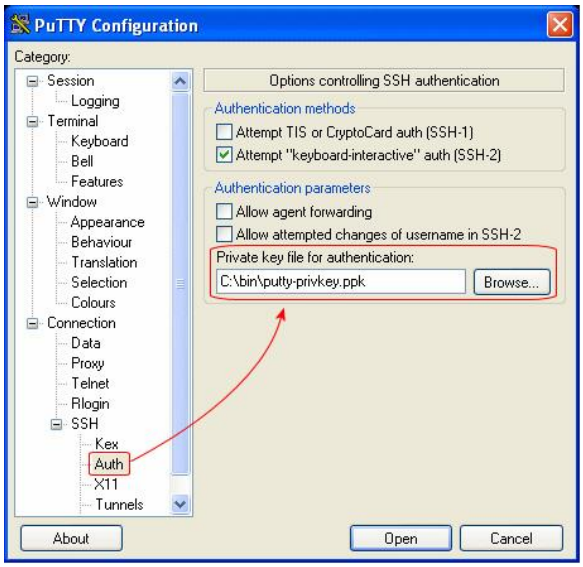


Рисунок 2.10 – Прикріплення публічного ключа до сесії

**Підключення через публічний ключ**

Запустимо PuTTY з можливістю завантажити збережену сесію з приватним ключем (рис.2.11):

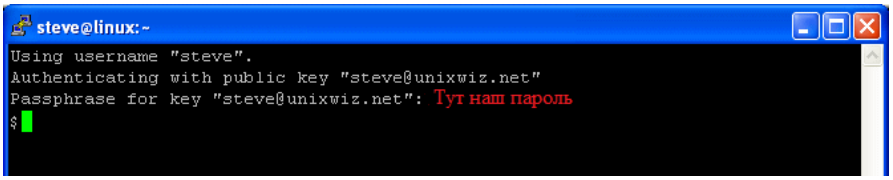


Рисунок 2.11 – Вікно запиту на введення парольної фрази

Замість того, щоб вводити пароль облікового запису (який відрізнятиметься на кожній віддаленій системі), система просить парольну фразу, яка захищає локальний приватний ключ. Коли приватний ключ вписується в публічний ключ на сервері OpenSSH, то дозволяє доступ і надає оболонку користувачеві.

Важливо відзначити, що при тому, що користувач повинен ввести секретне слово при вході в систему, фраза пов'язана з локальним приватним ключем, а не з віддаленим обліковим записом. Навіть, якщо публічний ключ користувача встановлений на 1000 різних віддалених серверах, вимагається одна і та ж фраза для приватного ключа для всіх них. Це значно спрощує завдання запам'ятовування облікових даних для доступу і робить його більш безпечним.

**Відключення парольної аутентифікації на OpenSSH**

Після того, як публічний і приватний ключі користувача перевірені як правильні, можна повністю відключити парольну аутентифікацію. Це виключає всі можливі спроби підбору паролю і різко зростає безпека.

Тим не менше, для машин, які фізично не локальні, буде доцільним відкласти відключення аутентифікації, поки не стане зовсім зрозуміло, що ключ доступу працює нормально, особливо, коли є кілька користувачів. Після того, як парольна аутентифікації буде відключена, навіть пароль адміністратора не дозволить увійти в систему.

Ці нові властивості для публічного ключа доступу рекомендується дуже уважно протестувати.

Налаштуванні демона SSH можна знайти у файлі `sshd_config`, який часто зберігаються в директорії `/etc/ssh/`. В цьому текстовому файлі, який відносно легко прочитати, ми шукатимемо два записи, які треба змінити.

Перший полягає у створенні для **PasswordAuthentication** значення **no**. Воно може бути явно встановлене в **yes**, або бути закоментоване, щоб використати значення за замовчуванням, але ми хочемо явно відключити його.

По-друге, ми хочемо відключити SSH протокол версії 1: він старий, має кілька суттєвих слабкостей в безпеці і не повинен бути дозволений для зовнішнього використання.

Змінюємо файл конфігурації і переконуємося, що два записи ключових слів встановлені правильно, а старі записи у файлі `/etc/ssh/sshd_config` закоментовані:

```
# Protocol1,2
Protocol 2
PasswordAuthentication no
```

Як тільки файл конфігурації буде збережений, демон безпечної оболонки повинен бути перезапущений; на більшості платформ це може бути зроблено з механізмом `service`:

```
# service sshd restart
```

Ми вбиваємо демона прослуховування і перезапускаємо його, але не припиняємо будь-які існуючі індивідуальні сесії користувача. Тим, хто вважають це ризикованим кроком, пропонується просто перезавантажити машину.

На даний момент, OpenSSH більше не прийматиме паролі будь-якого виду, доступ надається тільки для користувачів із заздалегідь встановленими публічними ключами.

### Дозвіл підтримки агента SSH

До цього моменту, ми приділили велику увагу безпеці доступу до системи, але він, як і раніше, не дуже зручний: ми, як і раніше, щоразу повинні вводити комплексну парольну фразу. Це може бути утомливо, якщо використовується велика кількість систем.

На щастя, набір SSH забезпечує прекрасний механізм для розблокування приватного ключа один раз, і дозволяє окремі з'єднання SSH без запиту на введення пароля щоразу.

Знайдемо і запустимо програму Pageant.exe з того ж місця, що й інші пов'язані з PuTTY файли, а він поставить себе в системному треї (у правому нижньому кутку поруч з годинником) (рис.2.12).

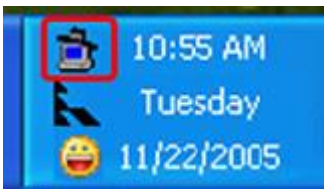


Рисунок 2.12 – Значок агента SSH

Двічі клацнемо на іконці в треї і запуститься діалогове вікно з порожнім списком ключів. Виберемо пункт **Додати ключ** (AddKey)і перейдемо в каталог **.ppk** файла, який містить приватний ключ. Коли буде запит на введення пароля, введемо його і натиснемо кнопку **ОК**. Натиснемо кнопку **Закрити** (Close), щоб закрити агента.

Тепер запустимо одну з уже налаштованої сесій SSH до віддаленого хоста, який використовує публічний ключ: він запитає агента для приватного ключа, віддалено обміняється з ним і надасть доступ без подальшого втручання користувача.

**Попереднє завантаження приватного ключа**

Перше, що багато користувачів PuTTY робить, коли входить в систему протягом дня, є запуск агенту і додавання приватного ключа. Це всього декілька кроків, але можемо оптимізувати їх трохи більше. Якщо запустимо агент з файлом приватного ключа як параметром, то він завантажуватиме ключ автоматично.

Перейдемо до Pageant.exe і клацнемо правою кнопкою миші, щоб скопіювати його значок. Вставимо значок як ярлик на робочому столі, потім клацнемо правою кнопкою миші і виберемо **Властивості**. Введемо повний шлях до **.ppk** файлу приватного ключа як параметру і збережемо зміни (рис.2.13).


Рис.2.13 – Налаштування попереднього завантаження ключа

Двічі клацнувши цей значок, ми будемо завантажувати файл ключів, який вимагає парольну фразу. Ввівши її один раз – це буде востаннє, коли це необхідно зробити, поки агент залишатиметься прикріпленим.

### 2.2. Завдання

1. Встановити та налаштувати на ПК з ОС Windows з'єднання SSH для віддаленого підключення до сервера з ОС на ядрі Linux.

2. Зробити і зберегти скріншоти виконання роботи.

### 2.3. Зміст звіту

1. Скріншоти з коментарями.

2. Відповіді на контрольні питання до роботи.

3. Короткі висновки.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 2.4. Контрольні питання

1. Які стандарти використовуються для ключа шифрування?

2. Які права треба встановити на файл ключа?

3. Для чого рекомендується відключити парольну аутентифікацію після налаштування доступу через SSH?

4. Які функції виконує агент SSH?

## Лабораторна робота 3

## KALI LINUX НА RASPBERRY PI ЯК ІНСТРУМЕНТ ПЕНТЕСТЕРА

**Мета роботи**: Реалізувати на мікрокомп’ютері RPi інструмент пентестера, використовуючи Kali Linux.

**Зміст.** Розглядається процес встановлення та початкового налаштування Kali Linux. На окремих прикладах демонструється виконання задач тестування системи на проникнення.

### 3.1. Загальні відомості

Kali Linux – це дистрибутив Debian – похідної від Linux системи, призначений для цифрової криміналістики і тестування на проникнення.

Поєднання Kali Linux з мікрокомп’ютером RPi додає нові можливості. Ключем до такої можливості є масове співтовариство розробників та виробників, які вносять тисячі нововведень для платформ Kali Linux і RPi.

RPi має кілька унікальних особливостей, які роблять його потужним і легко доступним комплектом інструментів тестера вразливостей. Зокрема, RPi дешевий, крім того, RPi компактний: він невеликий, тонкий, його легко заховати. А завдяки ОС Kali Linux, він отримує можливість запуску широкого спектру інструментів тестування від клонованих значків до скриптів крекінгу Wi-Fi. Замінивши SD-карту, або додавши чи видаливши компоненти, можна налаштувати RPi відповідно до будь-якої ситуації.

По-перше, RPi не суперкомп'ютер і не має величезної обчислювальної потужності. Тому, не дуже добре підходить для завдань з такими інтенсивними процесами, як перевірка паролів WPA їх перебором, або організація мережевих атак, бо з'єднання є занадто повільним. Але, RPi ідеально підходить для багатьох середовищ моделювання атаки. Ми просто перекладаємо перераховані завдання на потужні комп'ютери, а RPi використовуємо для збору даних.

RPi працює виключно добре, як платформа для моделювання атаки на Wi-Fi. Завдяки своєму невеликому розміру і великій бібліотеці інструментів атаки на базі Kali Linux, він ідеально підходить для розвідки і моделювання атак мереж

Wi-Fi. Наша збірка Kali Linux для тестування буде спрямована на анонімний польовий аудит дротових і бездротових мереж.

### 3.2. Налаштування Kali Linux на Raspberry Pi

Основні компоненти, необхідні для побудови системи тестування (атаки) з нашим RPi, і чому вони нам потрібні:

- Raspberry Pi. RPi 3 або 4 є платформою пропонованих збірок, координує і управляє всіма іншими компонентами. Низьке енергоспоживання і гнучкі можливості RPi дозволяють служити платформою для запуску інших операційних систем на базі Linux, крім Kali.
- Бездротова карта управління та контролю (C2). Метою бездротової карти C2 є автоматичне підключення RPi до керуючої AP, такої як телефонна точка доступу або домашня мережа. Це дозволяє здійснювати дистанційне керування RPi приховано або з великої відстані за допомогою SSH чи через VNC. RPi 3 має вбудовану карту Wi-Fi, але до RPi також може бути доданий адаптер бездротової мережі.
- Бездротова карта для тестування. Нашою бездротовою картою для моделювання атаки буде сумісний з Kali Linux Wi-Fi адаптер, здатний виконувати ін'єкції пакетів. Це буде наша площина атаки і може бути з дальнім або близьким радіусом дії, чи зі спрямованою антеною, в залежності від вимог поставленого завдання.
- Карти з записаними ОС. Хости ОС і мозок комп'ютера на мікро SD-карті можуть бути точно налаштовані для будь-якого бажаного середовища. Створивши налаштовані карти, можна швидко змінювати конфігурацію і функції RPi простою заміною карти і компонентів.
- Комп'ютер. Нам також знадобиться комп'ютер, щоб завантажити прошивку та записати її на мікроSD-карту.
- Електроживлення. RPi використовує стандартний блок живлення Micro-USB і майже будь-який зарядний пристрій для телефону або акумуляторна батарея будуть придатні для живлення RPi.

- Ethernet-кабель (додатково). Кабель Ethernet дозволяє обійти бездротову аутентифікацію шляхом прямої взаємодії з локальними мережами, до яких у нас є фізичний доступ. Такі спеціалізовані атаки, як PoisonTap також можуть скористатися інтерфейсом Ethernet для проникнення в комп'ютери.
- Bluetooth клавіатура (опціонально). Клавіатура Bluetooth корисна для взаємодії, коли є підключення HDMI.
- Корпус. Корпус потрібний кожному RPi, щоб захистити його.

В даній лабораторній роботі розглянемо два основні режими, в яких будемо працювати на RPi. У відкритій конфігурації RPi підключається до дисплея через кабель HDMI, а також приєднується бездротова миша і клавіатура. У тактичній конфігурації (рис.3.1) будемо використовувати ноутбук або смартфон, щоб отримати віддалений доступ до RPi через SSH. При підключенні RPi до точки доступу на своєму телефоні або до сусідньої дружньої AP, зможемо отримати доступ до RPi, залишаючись в змозі використовувати передані в полі дані.

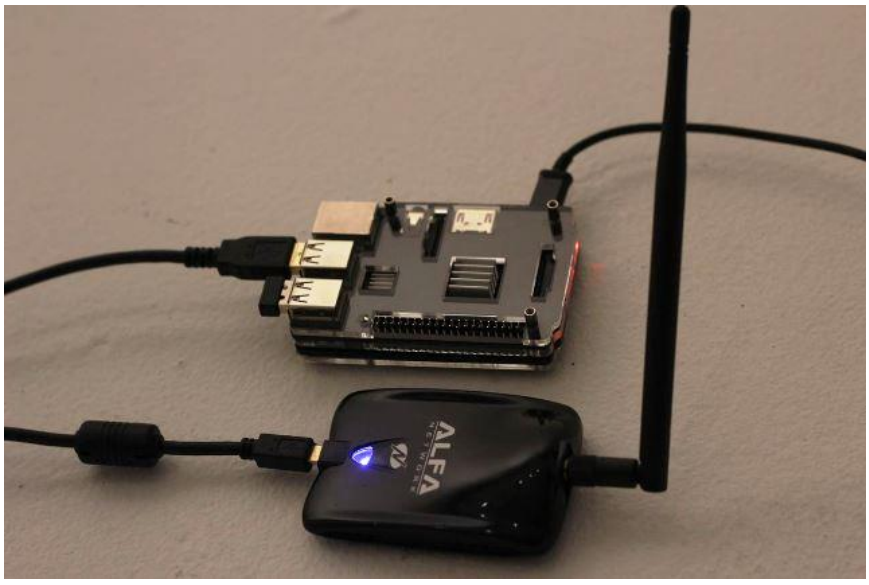


Рисунок 3.1 – Тактична конфігурація: Kali Linux через SSH

Розглянемо кроки, необхідні для налаштування RPi 3 як основної платформи з Kali Linux.

**Крок 1. Завантаження образу Kali Linux для Raspberry Pi**

Перейдемо на Offensive Security і завантажимо останню версію образу Kali Linux для RPi. Сьогодні це "RaspberryPi 2, 3, 4..." версії 2023.1 (рис.3.2).

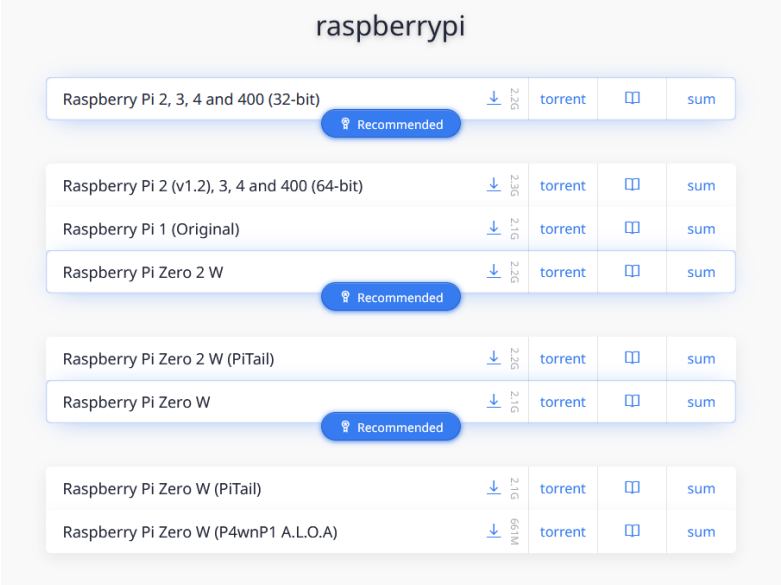


Рисунок 3.2 – Сторінка завантаження образу Kali Linux

**Крок 2. Запис образу на MicroSD карту**

Можемо використати такий інструмент, як ApplePiBaker for Mac або Etcher, щоб завантажити образ Kali на свою SD-карту, але іноді це може призвести до помилок. Щоб не допустити цього, уважно вивчіть, як записати образ на Mac за допомогою Terminal. Якщо використовуємо Windows, то можемо завантажити Win32 Disk Imager, щоб розмістити образ на карті.

**Крок 3. Завантаження в Kali Linux**

Коли закінчимо, карта SD буде готова для запуску. Вставляємо SD-карту в RPi, підключаємо монітор до HDMI, а також клавіатуру Bluetooth та джерело живлення, щоб вперше завантажитися в Kali Linux. Для того, щоб дістатися до робочого столу, логіном за замовчуванням є "`kali`", а паролем також "`kali`".

Процес залогінення є проблемою для автономного управління і нам треба буде його пізніше відключити. Це дозволить увімкнути RPi і відразу віддалено підключитися до нього без екрану (рис.3.3).

Рисунок 3.3 – Перше завантаження Kali Linux

**Крок 4. Оновлення Kali Linux**

Kali Linux – це особлива версія Debian Linux, призначена для тестування на проникнення. Вона сумісна з деякими з кращих і найсучасніших інструментів, доступних для бездротового пентеста і достатньо гнучка, щоб підтримувати велику кількість розробок для пентестерів. Вона підтримується Offensive Security і нам необхідно оновити її до останньої версії, щоб переконатися, що всі інструменти працюють належним чином.

Перш, ніж запустити, бажано розширити установку до розміру розділу. Для цього виконаємо команду:

```
resize2fs /dev/mmcblk0p2
```

У правому верхньому куті робочого столу побачимо можливість підключення до сусідньої бездротової мережі. Підключимося до точки доступу свого телефону або дружньої AP, щоб отримати оновлення. Виконаємо оновлення, відкривши вікно терміналу і ввівши наступні команди:

```
sudo apt update
sudo apt dist-upgrade -y
reboot
```

Kali встановиться в актуальний стан. Змінимо кореневий пароль на щось більш безпечне, ніж "`kali`", набравши:

```
passwd root
```

Потім введемо новий пароль для своєї системи Kali Linux.

**Крок 5. Установка сервера OpenSSH**

Для того, щоб спілкуватися зі своїм RPi з комп'ютера або телефону, ми повинні мати можливість увійти в систему. Для цього можемо використати SSH для підключення через будь-яке з'єднання з Wi-Fi, яке ділимо з RPi. SSH – мережевий протокол, який дозволяє виконувати команди на віддаленому пристрої. Це означає, що нам не потрібно підключати екран, щоб взаємодіяти з нашим RPi.

У терміналі, виконаємо наступні дії для встановлення сервера OpenSSH і оновлення рівнів запуску, щоб дозволити запуск SSH при завантаженні:

```
sudo apt install openssh-server
update-rc.d -f ssh remove
update-rc.d -f ssh defaults
```

Ключі за замовчуванням представляють величезну вразливість, так як будь-яка людина може вгадати їх. Давайте негайно змінимо їх, виконавши наступні команди:

```
cd /etc/ssh/
mkdir insecure_old
mv ssh_host* insecure_original_default_kali_keys/
dpkg-reconfigure openssh-server
```

Будуть створені резервні копії старих ключів SSH в іншій папці і згенеруються нові ключі. Проблема вирішена. Тепер давайте переконаємося, що можемо увійти через корінь ввівши:

```
sudo nano /etc/ssh/sshd_config
```

Ми відкрили для редагування папку конфігурації SSH. Змінимо рядок:

```
PermitRootLogin without-password
```

щоб він виглядав як:

```
PermitRootLogin yes
```

Натискаємо Ctrl+O, щоб зберегти зміни. Якщо все зроблено правильно, то нам не потрібно вже нічого змінювати.


Рисунок 3.4 – Налаштування `sshd_config`

Давайте перезапустимо службу SSH, ввівши команди:

```
sudo service ssh restart
update-rc.d -f ssh enable 2 3 4 5
```

І, нарешті, перевіримо, що SSH працює, за допомогою наступних дій, щоб побачити, чи запущена SSH в даний час (рис.3.5):

```
sudo service ssh status
```

Якщо щось не так, запустимо команду, щоб запустити вручну службу SSH:

```
sudo service ssh start
```


Рисунок 3.5 – Перевірка статуса SSH

Якщо виявимо, що SSH не працює, то можемо використати `raspi-config` як обхідний шлях. Команда буде працювати на Kali. Для того, щоб використовувати її, клонуємо з GitHub, наберемо `sudo mount /dev/mmcblk0p1 /boot` для монтування завантажувального розділу, перейдемо у директорію через `cd` і запустимо:

```
sudo bash raspi-config.
```

**Крок 6. Тестування входу за допомогою SSH**

Спробуємо увійти в систему з нашого домашнього комп'ютера або ноутбука. Підключимо RPi до тієї ж бездротової домашньої мережі або до робочої, до якої підключений комп'ютер. Виконаємо команду `ifconfig` на своєму RPi в терміналі, щоб дізнатися свою IP-адресу (рис.3.6):

```
ifconfig
```

На своєму персональному комп'ютері вводимо:

```
ssh root@(наша IP-адреса)
```


Рисунок 3.6 – IP виглядає як 10.11.1.144

Повинні побачити екран MOTD (рис.3.7).


Рисунок 3.7 – Простий MOTD при успішному SSH-вході в систему

Якщо немає, то можемо запустити Advanced IP Scanner на Windows, щоб побачити список усіх доступних пристроїв в мережі, якщо необхідно знайти IP-адресу свого RPi з персонального комп'ютера.

**Крок 7. Налаштування автологіну для віддаленого доступу**

Іноді, ми хочемо мати можливість увійти в інший обліковий запис, ніж кореневий. Давайте створимо нового користувача з ім'ям WHT (або будь-яким іншим) з кореневим дозволом за допомогою команди:

```
useradd -m WHT -G sudo -s /bin/bash
```

Змінимо пароль для `WHT` (або для того, як ми назвали) на щось більш безпечне, ніж "`kali`":

```
passwd WHT
```

Тепер відключимо цей вхід до того, поки зможемо завантажуватися безпосередньо в Kali, і наші бездротові карти запустяться і підключаться, щоб дозволити нам дистанційне керування. Для цього вводимо наступну команду:

```
sudo nano /etc/lightdm/lightdm.conf
```

І видаляємо знак # перед наступними рядками (рис.3.8):

```
autologin-user=root
autologin-user-timeout=0
```


Рисунок 3.8 – Організація автологіну

Збережемо і вийдемо за допомогою Ctrl+X. Далі вводимо:

```
sudo nano /etc/pam.d/lightdm-autologin
```

Треба змінити запуск в рядку 11:

```
# Allow access without authentication
auth required pam_succeed_if.so user != root quiet_success
auth required pam_permit.so
```

на такий (рис.3.9):

```
# Allow access without authentication
###auth required pam_succeed_if.so user != root quiet_success
auth required pam_permit.so
```

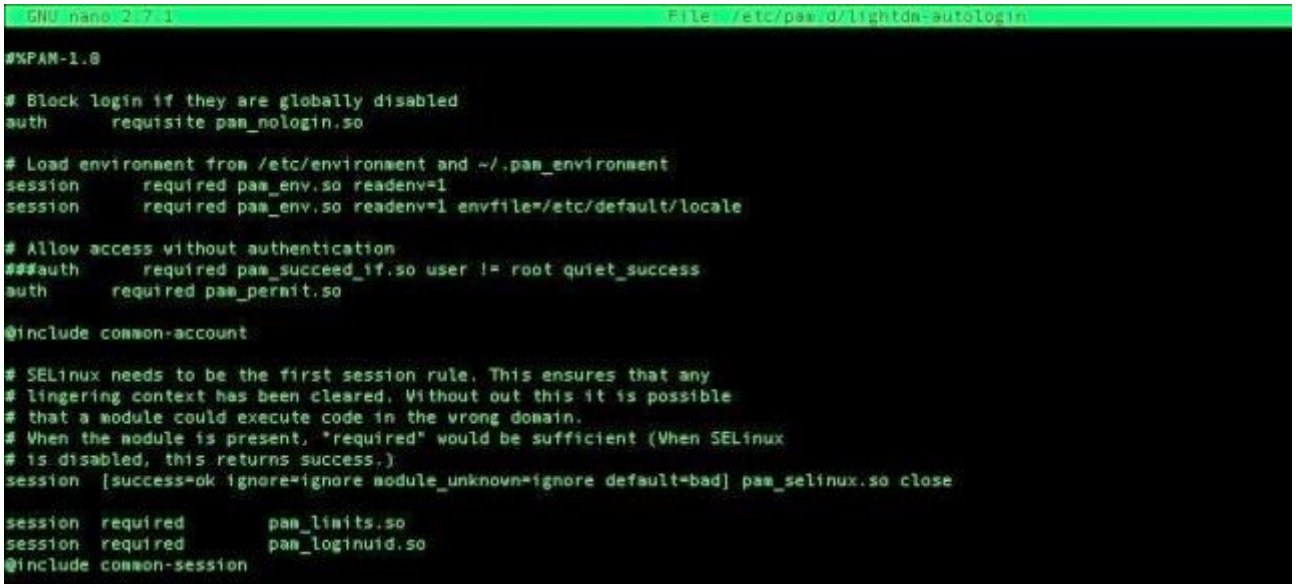


Рисунок 3.9 – Зміни у файлі `lightdm-autologin`

Збережемо і вийдемо з `nano`, далі вводимо `reboot` в терміналі, щоб перезапустити RPi для перевірки.

Для того, щоб вважати все готовим, пристрій повинен виконати наведений нижче контрольний список:

1. Пристрій запускається, входимо в систему без запиту пароля і запускається SSH при завантаженні, щоб дозволити віддалений доступ.
2. Пристрій підключається до команди AP, щоб включити дистанційне керування (робить це за замовчуванням після підключення в перший раз).
3. RPi може бути виключена без руйнування даних на мікро SD-карті (нормально завантажується після виключення).

Пройшли всі вимоги? Тоді наш RPi готовий продовжити роботу.

### 3.3. Приклади нструментів Kali Linux

Тепер ми готові спробувати інструменти Kali Linux, доступні відразу після встановлення. Доступних додатків так багато, що перевірити їх всі непосильно. Ми розглянемо кілька прикладів, які можемо легко спробувати.

**Змінюємо свою MAC-адресу**

MAC-адреса – це унікальний ідентифікатор для кожного мережевого адаптера. Вона залежить від кожного виробника, і часто використовується для надання доступу до певної частини мережі обмеженим комп'ютерам. Сервер DHCP також може призначати завжди ту саму IP-адресу MAC-адресу. Наприклад, ми можемо налаштувати свою мережу Wi-Fi, щоб додати нашу MAC-адресу в білий список і заборонити іншим підключатися до неї.

MacChanger – це інструмент, який дозволяє нам підмінювати MAC-адресу, тобто видавати себе за когось іншого.

Встановимо його, якщо потрібно:

```
sudo apt install macchanger
```

Переглянемо свою поточну MAC-адресу (рис.3.10):

```
ifconfig eth0
```

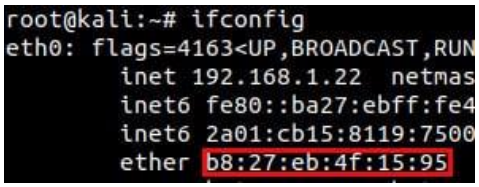


Рисунок 3.10 – Поточна MAC-адреса

Вимкнемо мережеву карту:

```
ifdown eth0
```

Отримаємо випадкову MAC-адресу:

```
macchanger -r eth0
```

Встановимо конкретну MAC-адресу:

```
macchanger -m XX:XX:XX:XX:XX:XX eth0
```

Перезавантажимо, щоб скинути налаштування та отримати стандартну MAC-адресу.

**Тестуємо надійність пароля Wi-Fi**

AirCrack-NG є одним із найпопулярніших інструментів у Kali Linux. Це повний набір інструментів для тестування безпеки бездротової мережі. Він надає

інструменти для моніторингу, атак, тестування та злому мереж Wi-Fi. Нам перед початком тестування треба відключити Wi-Fi на RPi.

Потім перевіряємо, чи наша мережева карта сумісна (рис.3.11):

```
sudo airmon-ng
```

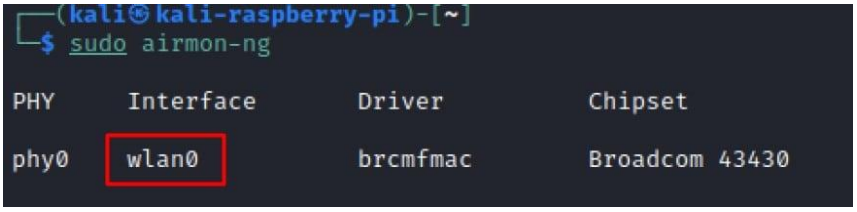


Рисунок 3.11 – Перевірка мережевої карти на сумісність

Запускаємо моніторинг:

```
sudo airmon-ng start wlan0
```

Переглядаємо доступні бездротові мережі (рис. 3.12):

```
sudo airodump-ng wlan0mon
```

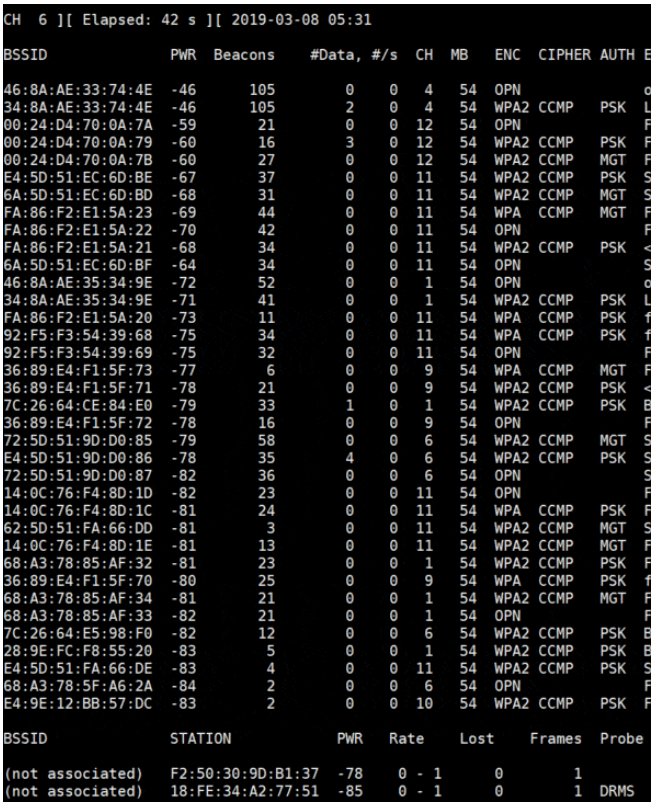


Рисунок 3.12 – Доступні бездротові мережі

Тепер ми готові працювати далі!

**Метод грубої сили з Hydra**

Груба сила (brute force) – це метод злому паролів, який намагається ввести паролі зі словника чи інших джерел і випробовує всі можливості, поки не спрацює. Hydra – це інструмент із програмного забезпечення Kali Linux для дуже швидкого перебору даних, який підтримує багато протоколів.

Для використання нам знадобиться список паролів, який помістимо у файл, наприклад /root/passwords.txt (по одному в рядку). Ми можемо знайти найпоширеніші паролі в Інтернеті або створити власні. Для перевірки просто вручну додаємо кілька випадкових паролів у файл. Тоді зможемо спробувати, наприклад, перебороти SSH на своєму комп'ютері з RPi:

```
hydra -l root -P /root/passwords.txt -t 6 ssh://192.168.222.51
```

Якщо перевіримо свій `/var/log/auth.log`, то побачимо спроби з RPi:

```
May 22 15:55:37 ubuntu sshd[2481]: Failed password for root from
192.168.222.31 port 37226 ssh2
May 22 15:55:37 ubuntu sshd[2487]: Failed password for root from
192.168.222.31 port 37234 ssh2
May 22 15:55:39 ubuntu sshd[2482]: Failed password for root from
192.168.222.31 port 37228 ssh2
May 22 15:55:39 ubuntu sshd[2484]: Failed password for root from
192.168.222.31 port 37232 ssh2
```

**Аналізатор пакетів**

Аналізатор пакетів (або сніффер) – це інструмент, який може перехоплювати трафік із мережі та фіксувати його для аналізу. У Kali Linux ми можемо використовувати Wireshark, який є найбільш розповсюдженим інструментом для аналізу мережевого трафіку. Це графічний інструмент, але можемо захоплювати пакети за допомогою `tcpdump` або чогось іншого, а потім відкривати їх за допомогою Wireshark.

Програму можна знайти в меню Applications в розділі Sniffing and spoofing:

- Запустимо його, а потім перейдемо до Capture > Start.
- Побачимо всі пакети з мережі (рис.3.13).
- Натиснемо Stop, коли захочемо.

Крім того, є багато функцій, за допомогою яких можна фільтрувати чи аналізувати те, що ми зафіксували:

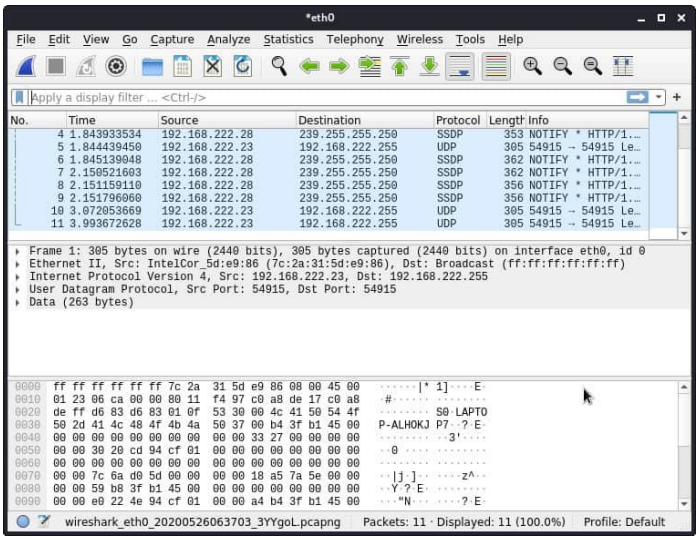


Рисунок 3.13 – Захоплені з мережі пакети

**SQL ін'єкція**

SQL-ін’єкція – це техніка атаки на незахищені програми, включаючи введення коду в незахищені поля користувача. Ця техніка в основному використовується для атак на веб-сайти. Наприклад, якщо ми замінюємо параметр URL-адреси, скажімо, *?user=наше ім’я* на щось на зразок *?user=наше ім’я ‘ OR 1*. Якщо поле погано захищене, SQL-запит буде змінено та поверне всі дані, а не лише дані нашого користувача. У Kali Linux інструмент `sqlmap` дозволяє тестувати вразливості ін’єкції SQL.

Sqlmap – простий у використанні інструмент. Нам потрібно лише вказати URL-адресу сторінки для перевірки, приблизно так:

```
sqlmap -u https://www.domain.com/?p=123
```

Коли ми знайшли проблему в безпеці, то можемо копнути глибше за допомогою цього інструменту, щоб побачити, що ще можемо отримати. Але найкраще, що можна зробити, це виправити вразливість.

**Експлойт уразливостей**

Metasploit – це інструмент, який дозволить нам перевіряти вразливості та використовувати їх. Metasploit дозволяє автоматизувати процес виявлення та

використання, а також надає інструменти, необхідні для виконання етапу тестування на проникнення вручну.

Ми можемо запустити його в Applications > Exploitation Tools > Metasploit framework. Цей інструмент ініціалізує та запустить термінал, який дозволить нам його використовувати.

Наприклад, можемо використати `nmap` у фреймворку (рис.3.14):

```
db_nmap -v -sV 192.168.222.1
```

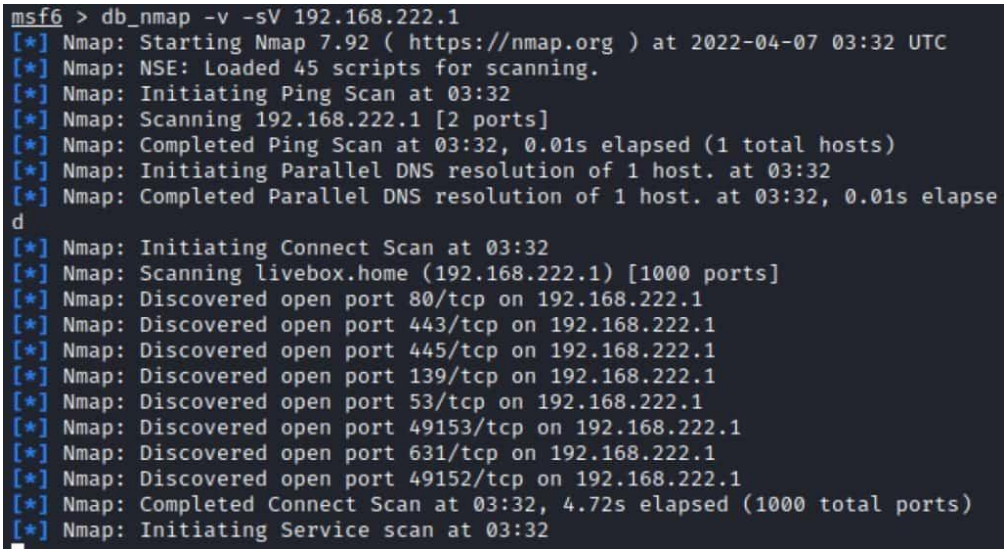


Рисунок 3.14 – Використання фреймворком Metasploit утиліти `nmap`

Ми також можемо отримати інформацію про відому вразливість і спробувати її використати:

```
db_rebuild_cache
search CVE-2018-9864
use exploit/folder/folder/name
```

Замінимо параметр пошуку своїм ідентифікатором уразливості та використовуємо шлях до експлойту, який відображається в результатах пошуку.

***Увага!*** *Задачі пентестера не повинні виходити за рамки закону і виконуватися лише при наявності офіційного доступу до мережі, яка тестується.*

### 3.4. Завдання

1. Встановити, як описано, Kali Linux на RPi, або іншій ОС на ядрі Debian. Для Windows необхідно попередньо встановити віртуальну машину.

2. Оновити встановлену систему, щоб завантажились необхідні утиліти для виконання задач пентестера (в меню буде доступно біля 300 утиліт).

3. Знайти в меню та коротко описати в протоколі ЛР призначення утиліт для тестування радіоканалів (WiFi, RFID тощо).

### 3.5. Зміст звіту

1. Скріншоти виконання завдання.

2. Відповіді на контрольні питання до роботи.

3. Короткі висновки.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 3.6. Контрольні питання

1. Для чого розширюють установку до розміру розділу на карті?

2. Для чого при тестуванні бажано змінити MAC-адресу?

3. З якими параметрами можна запустити AirCrack-NG?

## Лабораторна робота 4

# ВИКОРИСТАННЯ WIRESHARK ДЛЯ АНАЛІЗУ ТРАФІКА

**Мета роботи:** Метою даного заняття є придбання навичок захоплення мережевого трафіку в сегменті локальної мережі та аналізу зібраної інформації за допомогою програмного аналізатора протоколів Wireshark.

**Зміст.** Розглядаються налаштування Wireshark, побудова та використання фільтрів при аналізі трафіку.

## 4.1. Загальні відомості

Wireshark – це потужний мережевий аналізатор, який може використовуватися для аналізу трафіку, що проходить через мережевий інтерфейс нашого комп’ютера. Він може знадобитись для виявлення і вирішення проблем з мережею, налаштування наших веб-додатків, мережевих програм або сайтів, виявлення та вирішення проблем у забезпеченні безпеки вузлів комп'ютерної мережі та інформації, що циркулює між ними. Wireshark дозволяє повністю передивлятися вміст пакету на всіх рівнях: так ми можемо краще зрозуміти як працює мережа на низькому рівні.

Розглянемо більш детально, які можливості підтримує програма, з якими протоколами вона може працювати. Основні можливості програми:

- Захоплення пакетів в реальному часі з дротового або будь-якого іншого типу мережевих інтерфейсів, а також читання з файла.
- Підтримуються такі інтерфейси захоплення: Ethernet, IEEE 802.11, PPP і локальні віртуальні інтерфейси.
- Пакети можна відсіяти за багатьма параметрами за допомогою фільтрів.
- Всі відомі протоколи підсвічуються у списку різними кольорами, наприклад, TCP, HTTP, FTP, DNS, ICMP тощо.
- Підтримка захоплення трафіку VoIP-дзвінків.

- Підтримується розшифровка HTTPS-трафіку при наявності сертифіката.
- Розшифровка WEP, WPA-трафіку бездротових мереж при наявності ключа і handshake.
- Відображення статистики навантаження на мережу.
- Перегляд вмісту пакетів для всіх мережевих рівнів.
- Відображення часу надсилання і отримання пакетів.

Програма має множину інших функцій, а вище перераховані лише основні, які можуть нас зацікавити.

**Як користуватися Wireshark**

Якщо програма ще не встановлена, то її можна встановити з офіційних репозиторіїв. Для Windows скористаємося посиланням https://www.wireshark.org/download.html (рис.4.1).

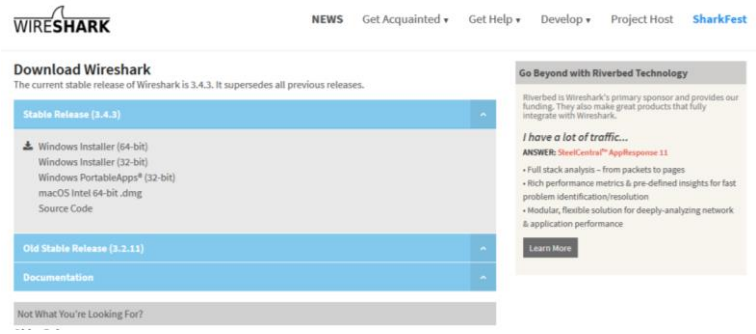


Рисунок 4.1 – Сторінка завантаження Wireshark

Головне вікно програми показане на рис.4.2.

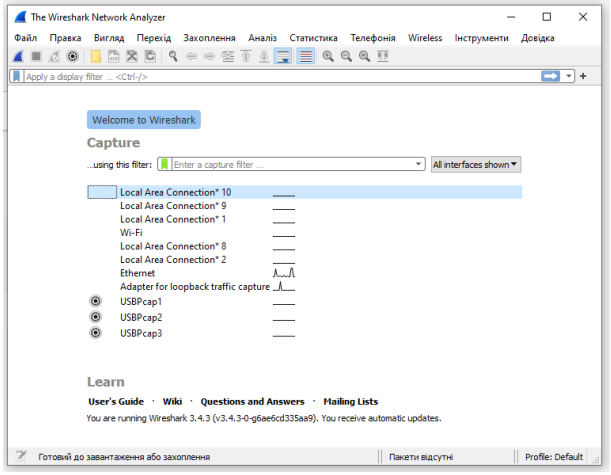


Рисунок 4.2 – Головне вікно програми

Це вікно відкривається при запуску програми і у ньому відображаються мережеві з’єднання (адаптери) (рис.4.2).

Виберемо одне із з’єднань, наприклад, Ethernet, і натиснемо крайню (ліворуч під пунктом меню «Файл») кнопку **«Почати захоплення пакетів»** для захоплення пакетів.

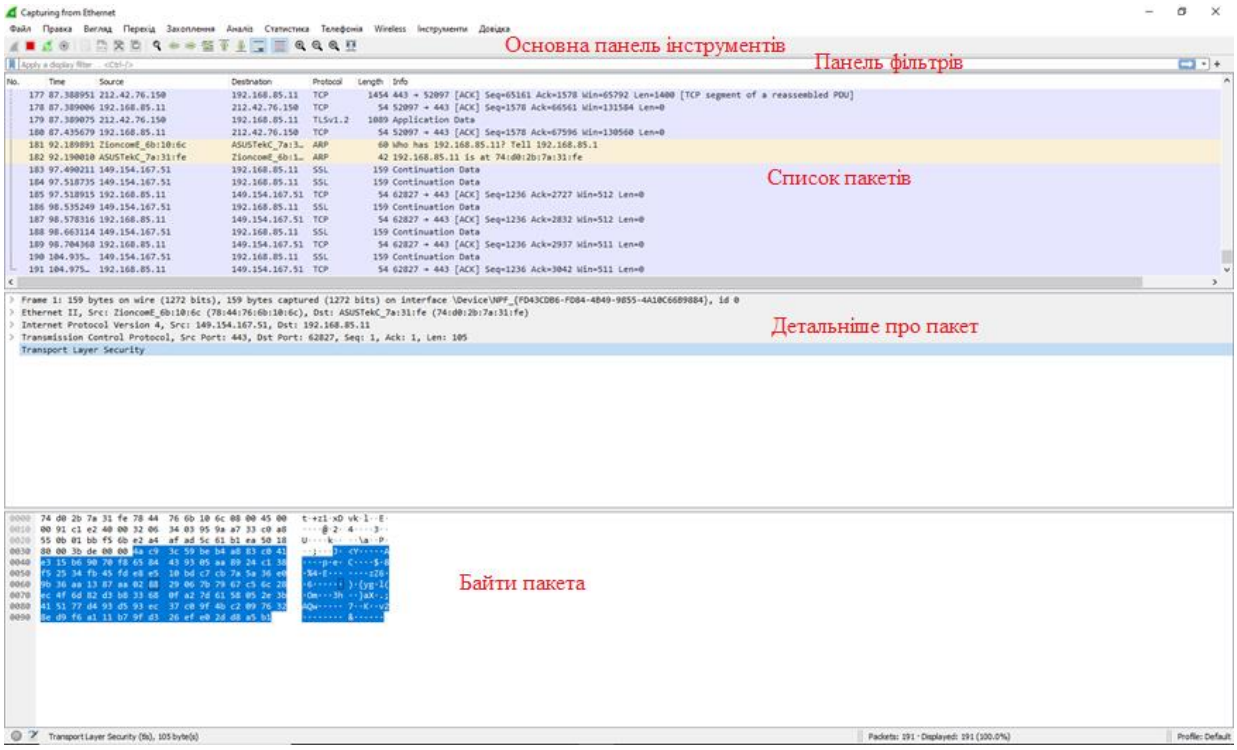


Рисунок 4.3 – Захоплені пакети з’єднання Ethernet

Переглянемо коротко елементи меню (рис.4.3):

- Вверху, під головним меню, знаходиться панель з кнопками – це основна панель інструментів (**Main Toolbar**) на якій розміщені основні елементи керування програмою – старт/стоп, налаштування захоплення, відкриття/закриття файлів захоплення, навігація пакетами та керування розміром тексту.
- Відразу під цією панеллю знаходиться панель фільтрів (**Filter Toolbar**) – тут будемо записувати фільтри і оператори, які розглянемо пізніше.
- Ще нижче знаходиться список пакетів (**Packet List**) – це таблиця, в якій відображаються всі пакети поточної сесії захоплення або з відкритого файлу захоплення.

- Під ним – детальна інформація про пакет (**Packet Details**) – тут відображаються відомості про вибраний пакет (вибирати пакети можна в Packet List).
- Найнижче вікно – байти пакета (**Packet Bytes**) – тут показані первинні дані пакета в необробленому вигляді, тобто, в тому вигляді, в якому пакет передається в мережі.

Для початку аналізу виберемо мережевий інтерфейс, на якому виконаємо захоплення пакетів**.**

Далі можемо клацнути на будь-якому пакеті (рис.4.4), щоб зробити його аналіз.

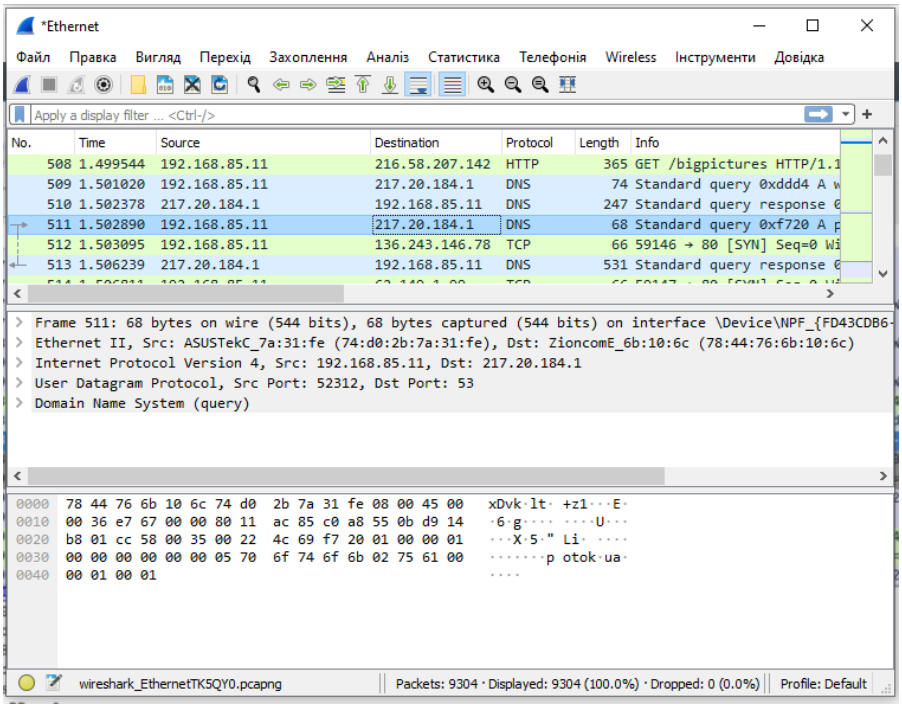


Рисунок 4.4 – Виділений пакет запиту до DNS

На рис.4.4 бачимо пакет запиту до DNS, щоб отримати IP-адресу сайта: в самому запиті надсилається домен, а в пакеті відповіді отримуємо наше питання, а також відповідь.

Для більш зручного перегляду можна відкрити пакет в новому вікні, виконавши подвійне клацання на записі (рис.4.5):

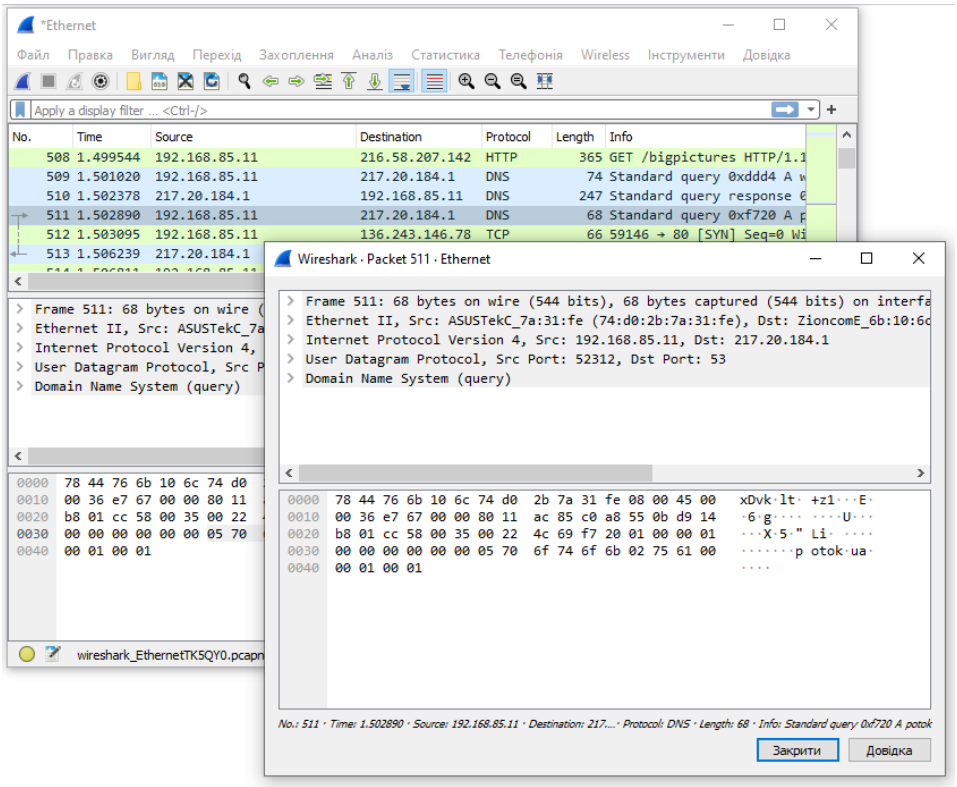


Рисунок 4. 5. Перегляд пакета в окремому вікні

## 4.2. Фільтри Wireshark

Перебирати пакети вручну, щоб знайти необхідні, при активному потоці незручно. Тому для такої задачі краще використовувати фільтри. Їх у Wireshark два типа: capture і display – для захоплення вже конкретних мережевих пакетів (рис.4.6 і 4.7) і для відображення визначених пакетів із захопленої різнорідної маси пакетів (рис.4.8), відповідно.

Якщо ми наперед знаємо які пакети нам треба, то можемо налаштувати фільтр захоплення, відразу вказавши програмі, що зберігати, а що відкидати, тобто, це своєрідний вхідний фільтр в програму.

*Плюси:* Економія місця на диску, якщо доводиться зберігати великий мережевий дамп.

*Мінуси:* Навантажує процесор. Використовується лише для живого сніфингу трафіка.

Фільтр відображення використовується, коли у нас вже є пакети різних протоколів і хостів, отримані в процесі живого сніфингу або збережені на

комп’ютері. Даний тип фільтра представляє собою свого роду вихідний фільтр, бо в програму вже потрапили різні пакети і треба з цієї маси виділити щось конкретне.

*Плюси:* Можна фільтрувати онлайн трафік і оффлайн файли.

*Мінуси:* При онлайн захопленні на комп’ютері зберігається велика кількість зайвих пакетів.

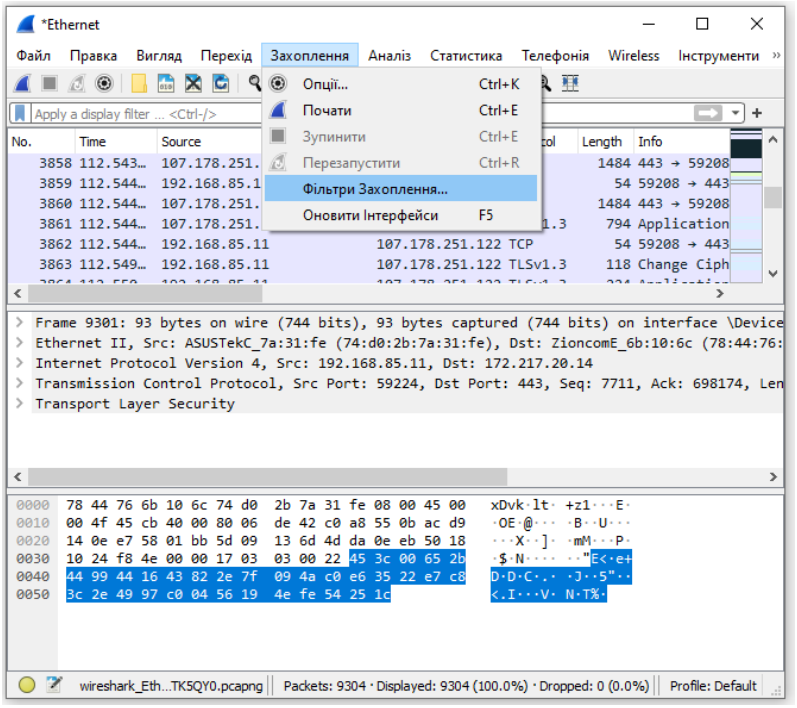


Рисунок 4.6 – Меню вибору і створення фільтрів захоплення

Мова фільтрів захоплення стандартна для Open Source і використовується багатьма продуктами на основі Pcap (наприклад, утилітою tcpdump або системою виявлення/запобігання вторгнень Snort). Тому детально описувати синтаксис не будемо. При бажанні, його можна переглянути в документації, наприклад, в Linux на сторінці довідника про pcap-filter.

Фільтри відображення, які працюють із вже перехопленим трафіком, є «рідними» для Wireshark. Їх відмінність від Pcap у форматі запису (наприклад, як роздільники полів використовується крапка); також додані англійська нотація в операціях порівняння і підтримка підрядків.

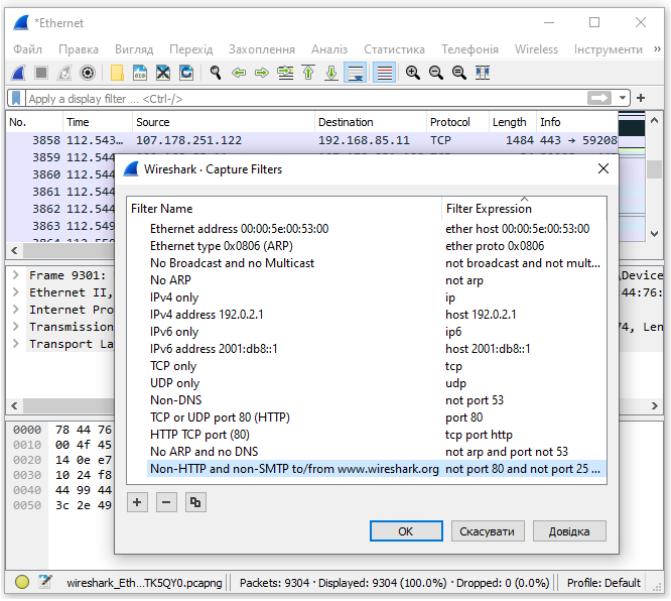


Рисунок 4.7 – Список існуючих фільтрів захоплення і кнопка (+) створення нового фільтра

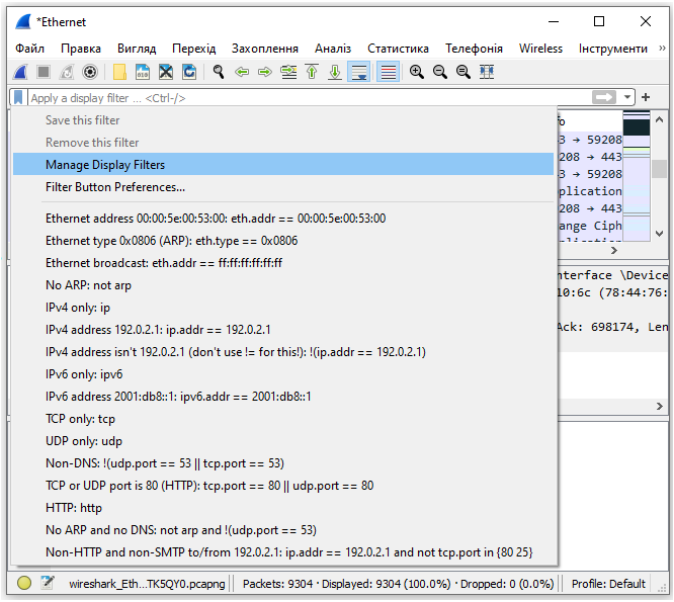


Рисунок 4.8 – Меню керування фільтрами відображення

До основних фільтрів належать:

- **ip.dst** – цільова IP-адреса;
- **ip.src** – IP-адреса відправника;
- **ip.addr** – IP відправника або отримувача;
- **ip.proto** – протокол;
- **tcp.dstport** – порт призначення;
- **tcp.srcport** – порт відправника;

- **ip.ttl** – фільтр по ttl, визначає мережеву відстань;
- **http.request_uri** – запитувана адреса сайту.

Найбільш часто використовувані вирази фільтрації та їх значення наведені в табл.4.1.

Таблиця 4.1 – Вирази для фільтрації та їх значення

| Вираз | Значення виразу і приклад запису |
|---|---|
| frame.marked | Маркований кадр: frame.marked == true |
| frame.number | Номер кадра: frame.number == 150 |
| frame.time | Час захоплення кадра: frame.time == "Feb 1, 2006 09:00:00" |
| frame.pktlen | Довжина кадра: frame.pkt_len == 48 |
| eth.dst | Заголовок Ethernet – MAC-адреса призначення: eth.dst == |
| eth.src | Заголовок Ethernet – MAC-адреса джерела: eth.src == 00:a0:cc:30:c8:db |
| eth.type | Заголовок Ethernet – тип вкладеного протоколу: eth.type == 0x0800 |
| arp.hw.type | Заголовок протоколу ARP – тип протоколу канального рівня: arp.hw.type |
| arp.proto.type | Заголовок протоколу ARP – тип протоколу мережевого рівня: arp.proto.type |
| arp.opcode | Заголовок протоколу ARP – код операції: arp.opcode == 0x0001 |
| arp.src.hwmac | Заголовок протоколу ARP – MAC-адреса джерела: arp.src.hw_mac == |
| arp.src.proto_ipv | Заголовок протоколу ARP – IP-адреса джерела: arp.src.proto_ipv4 == |
| arp.dst.hwmac | Заголовок протоколу ARP – MAC-адреса призначення: arp.dst.hw_mac == |
| arp.dst.proto_ipv | Заголовок протоколу ARP – IP-адреса призначення: arp.dst.proto_ipv4 == |
| ip.version | Заголовок протоколу IP - версія протоколу IP: ip.version == 4 |
| ip.hdrlen | Заголовок протоколу IP - довжина заголовка: ip.hdr_len == 24 |
| ip.flags.df | Заголовок протоколу IP - прапор фрагментації: ip.flags.df == 0 |
| ip.flags.mf | Заголовок протоколу IP - прапор не останнього фрагменту: ip.flags.mf == 0 |
| ip.fragoffset | Заголовок протоколу IP - зміщення фрагмента: ip.frag_offset == 0 |
| ip.ttl | Заголовок протоколу IP - час життя пакета: ip.ttl == 1 |
| ip.proto | Заголовок протоколу IP - протокол більш високого рівня: ip.proto == 0x01 |
| ip.src | Заголовок протоколу IP: IP-адреса джерела: ip.src == 10.0.0.99 |
| ip.dst | Заголовок протокола IP - IP-адреса призначення: ip.dst == 224.0.0.2 |
| ip.addr | Заголовок протоколу IP: IP-адреса: ip.addr == 10.2.0.0/16 |
| tcp.srcport | Заголовок протоколу IP - порт джерела: tcp.srcport == 1054 |
| tcp.dstport | Заголовок протоколу IP - порт призначення: tcp.dstport == 21 |
| tcp.seq | Заголовок протоколу IP - послідовний номер: tcp.seq == 4856133 |
| tcp.ack | Заголовок протоколу IP - номер підтвердження: tcp.ack == 4856134 |
| tcp.flags.urg | Заголовок протоколу IP - біт присутності термінових даних: tcp.flags.urg == |
| tcp.flags.ack | Заголовок протоколу IP - біт присутності підтвердження: tcp.flags.ack == 1 |
| tcp.flags.push | Заголовок протоколу IP - біт виштовхування даних: tcp.flags.push == 0 |
| tcp.flags.reset | Заголовок протоколу IP - біт скидання з’єднання: tcp.flags.reset == 0 |
| tcp.flags.syn | Заголовок протоколу IP - біт синхронізації сесії: tcp.flags.syn == 1 |
| tcp.flags.fin | Заголовок протокола IP - біт завершення сесії: tcp.flags.fin == 0 |
| tcp.windowsize | Заголовок протоколу IP - розмір приймального вікна: tcp.window_size == |
| udp.srcport | Заголовок протоколу UDP - порт джерела: udp.srcport == 2364 |
| udp.dstport | Заголовок протоколу UDP - порт призначення: udp.dstport == 53 |
| icmp.type | Заголовок протоколу ICMP - тип повідомлення: icmp.type == 8 |
| icmp.code | Заголовок протоколу ICMP - уточнюючий код повідомлення: icmp.code == |

Щоб вказати відношення між полем і значенням у фільтрі можна використовувати такі оператори:

- == – рівне;
- **!=** – не рівне;
- < – менше;
- > – більше;
- <= – менше або рівне;
- >= – більше або рівне;
- **matches** – регулярний вираз;
- **contains** – має.

Для об’єднання кількох виразів можна застосувати:

- **&&** – обидва вирази повинні бути істинними для пакета;
- || – може бути істинним один з виразів.

Через меню **Аналіз** (рис.4.9) можна отримати доступ до конструктора фільтрів відображення (рис.4.10).

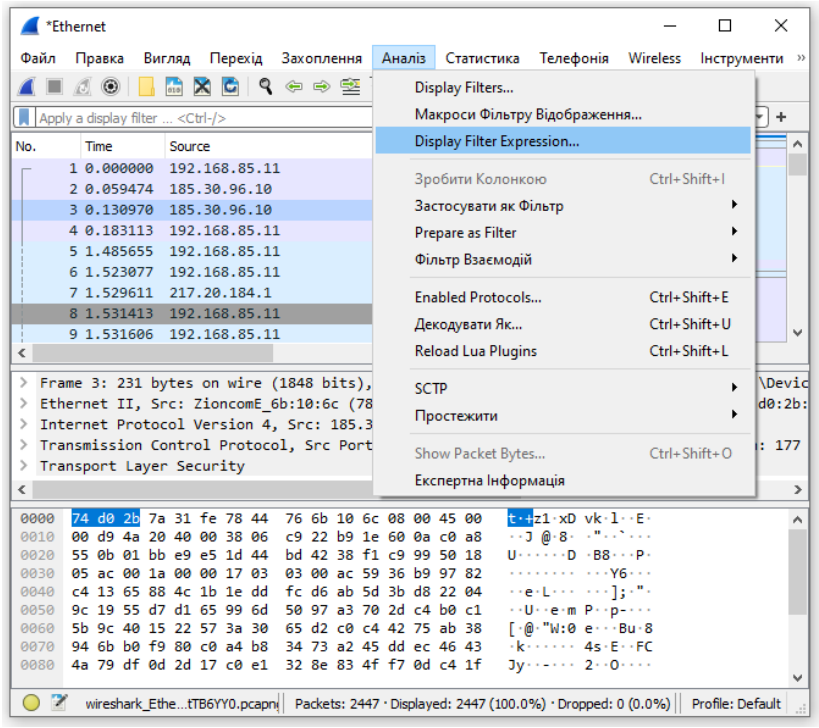


Рисунок 4.9 – Доступ до конструктора фільтрів відображення

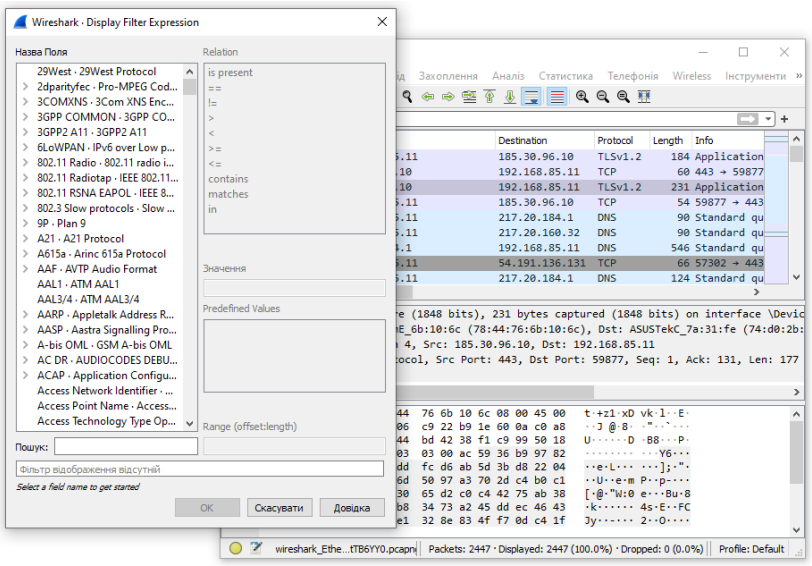


Рисунок 4.10 – Конструктор фільтрів відображення

Тепер розглянемо все більш детально на прикладах кількох фільтрів та спробуємо зрозуміти всі знаки відношень.

Спочатку відфільтруємо всі пакети, надіслані на 212.42.76.253 (ukr.net). Введемо рядок в поле фільтра (рис.4.11) і натиснемо **Apply** (стрілка праворуч в рядку фільтрів):

```
ip.dst == 212.42.76.253
```

Для зручності фільтри Wireshark можна зберегти за допомогою меню **Save this filter** (через піктограму ліворуч рядка фільтрів).

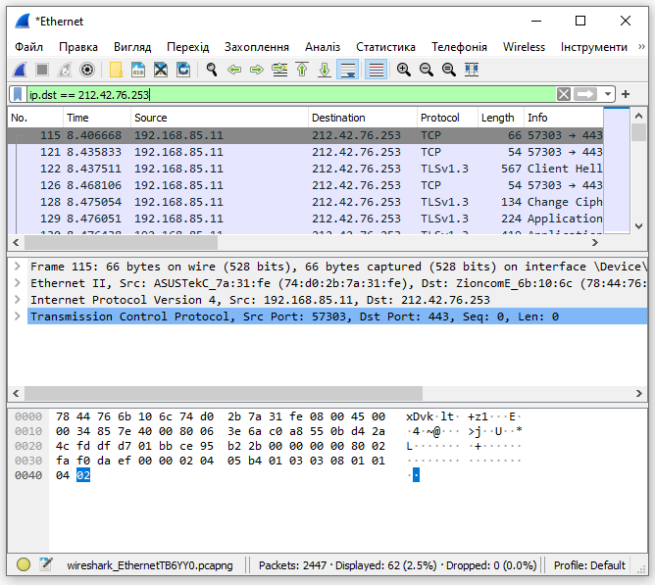


Рисунок 4.11 – Використання фільтра за IP-адресою отримувача

Щоб отримати не лише надіслані пакети, але і отримані у відповідь від цього вузла, можна об’єднати дві умови:

```
ip.dst == 212.42.76.253 || ip.src == 212.42.76.253
```

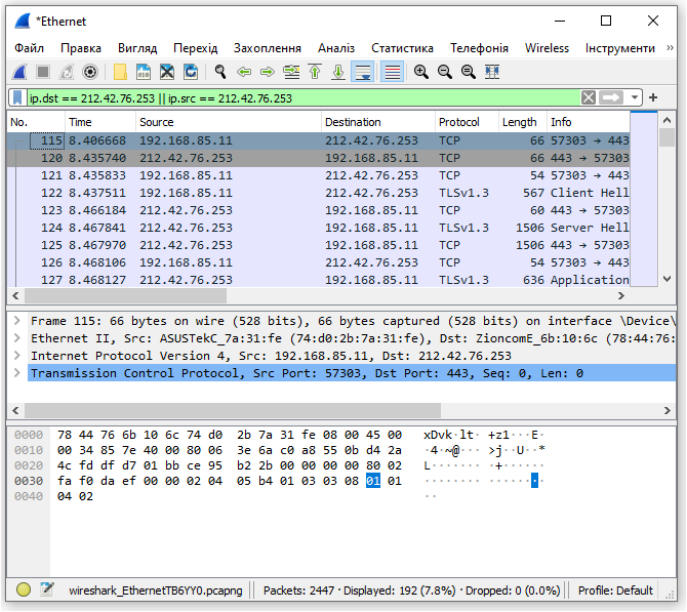


Рисунок 4.12 – Фільтр з двома умовами

Далі відберемо пакети з ttl меншим 10 (рис.4.13):

```
ip.ttl < 10
```

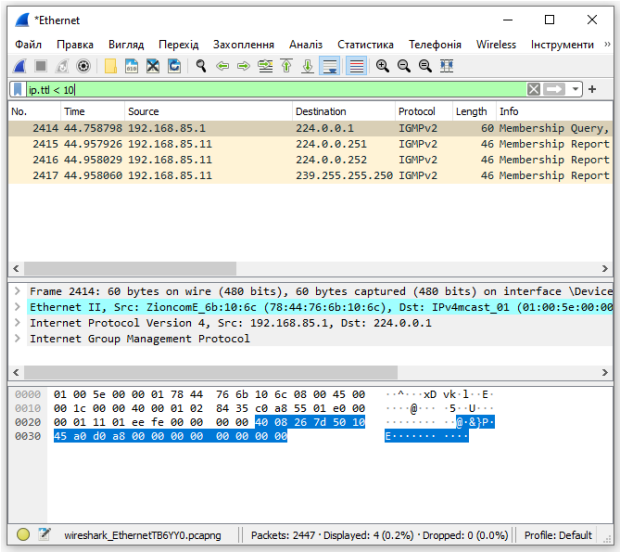


Рисунок 4.13 – Фільтр за значенням ttl

Також можемо відібрати передані великі файли:

```
http.content_length > 5000
```

Щоб очистити фільтр треба натиснути кнопку **Clear**. Буває, що не завжди відома вся необхідна для фільтрації інформація, а просто є необхідність дослідити мережу. Можемо додати будь-яке поле пакета в якості стовпця і подивитися його вміст в загальному вікні для кожного пакета.

Наприклад, бажано вивести у вигляді стовпця ttl (час життя) пакета. Для цього відкриємо інформацію про пакет, знайдемо це поле (Time to Life) в розділі IP (рис.4.14). Потім викличемо контекстне меню і виберемо опцію **Зробити Колонкою (Apply As Column)**:

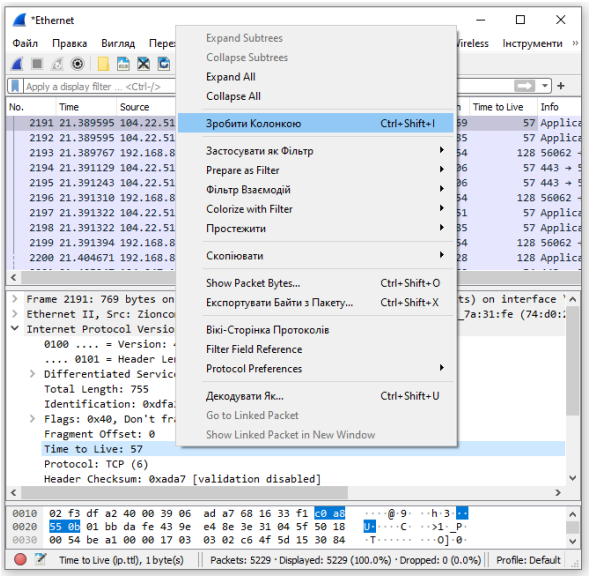


Рисунок 4.14 – Вибір опції **Зробити колонкою**

Далі після оновлення побачимо необхідний стовпець (рис.4.15):

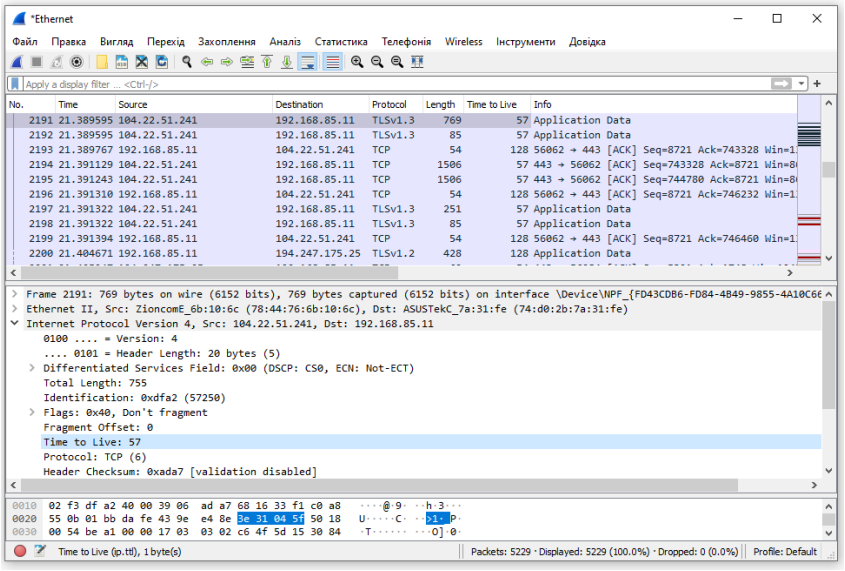


Рисунок 4.15 – Сповпець із значеннями ttl

Таким же чином можна створити фільтр на основі необхідного поля. Виберемо його і викликаємо контекстне меню, потім натиснемо **Застосувати як фільтр (Apply as filter)** або **Prepare as filter**. Далі вибираємо **Selected,** щоб вивести лише обрані значення, або **Not selected**, щоб їх видалити (рис.4.16):

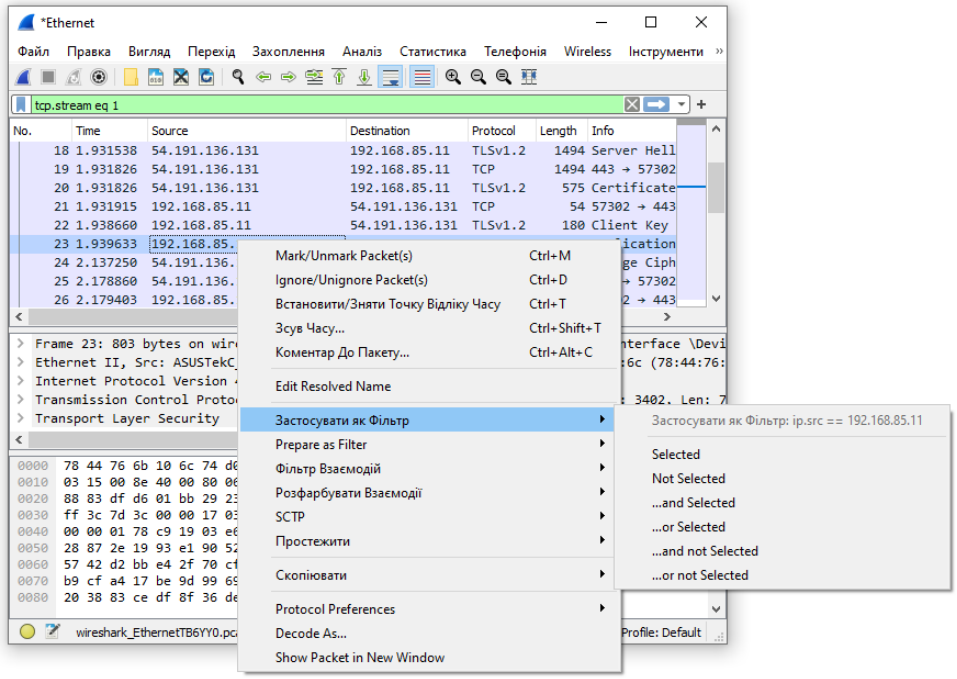


Рисунок 4.16 – Створення фільтра на основі вибраного поля

Вказане поле і його значення будуть застосовані або в другому випадку підставлені в поле фільтра:

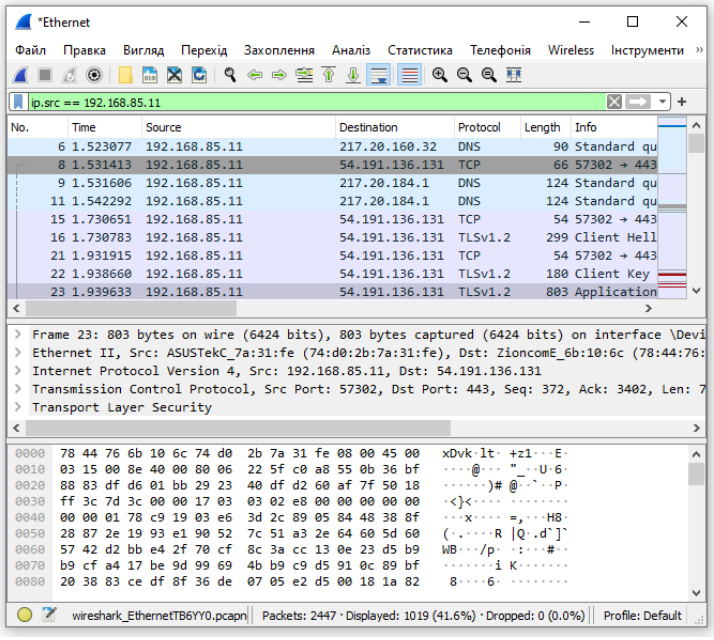


Рисунок 4.17 – Фільтрація за вибраним полем

Таким способом можна додати у фільтр поле будь-якого пакета або стовпець. Також ця опція є в контекстному меню. Для фільтрації протоколів можна використати і більш прості умови. Наприклад, виконати аналіз трафіку Wireshark для протоколів HTTP і DNS:

```
http || dns
```

Ще одна цікава можливість програми – використання Wireshark для відстеження певного сеансу між комп'ютером користувача і сервером. Для цього відкриваємо контекстне меню для пакета і вибираємо **Простежити – TCP-потік (Follow – TCP stream)** (рис.4.18).

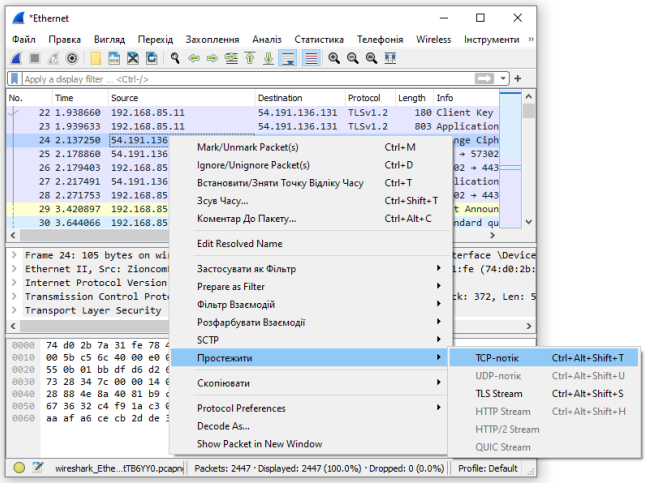


Рисунок 4.18 – Відстеження TCP-потоку

Потім відкриється вікно (рис.4.19), в якому знайдемо всі дані, які були передані між сервером і клієнтом.

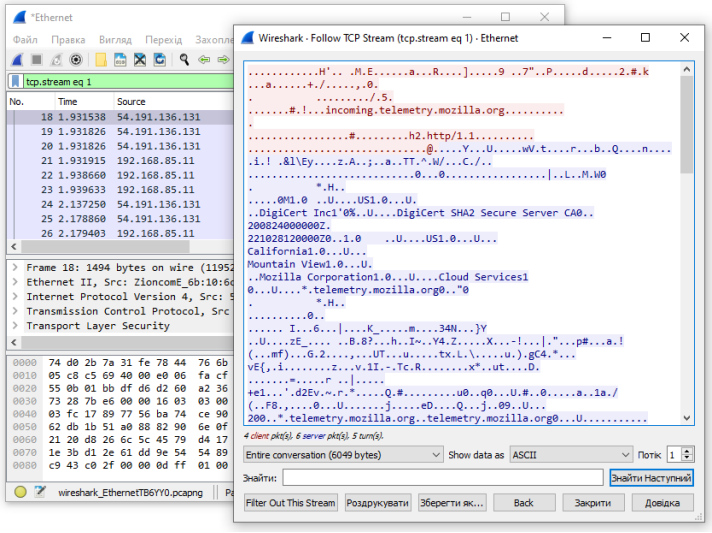


Рисунок 4.19 – Вікно обміну даними між сервером і клієнтом

### 4.3. Аналіз трафіку з Wireshark

З Wireshark можна легко зрозуміти, що саме завантажували користувачі і які файли вони дивились, якщо з'єднання не було зашифроване. Програма дуже добре виконує витягування контенту.

Для цього спочатку зупиняємо захоплення трафіку за допомогою червоного квадрата на панелі. Потім відкриваємо меню **File** -> **Export Objects** -> **HTTP** (рис.4.21):

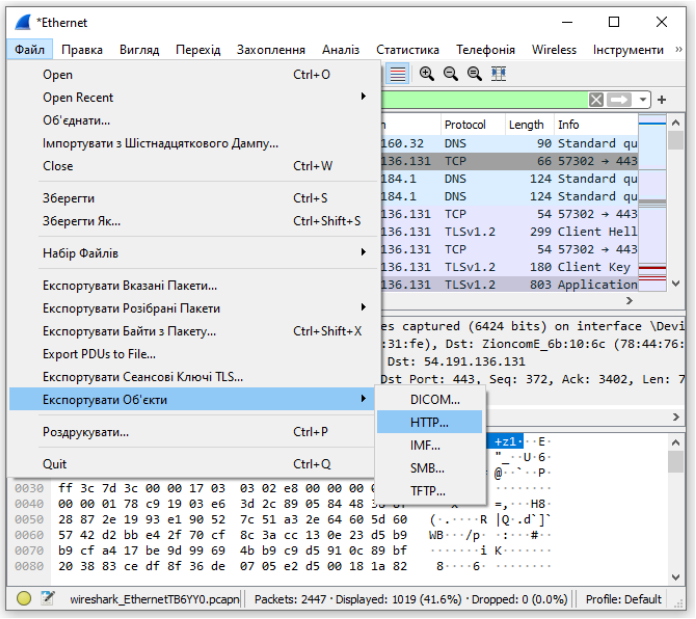


Рисунок 4.21 – Експорт вибраних об'єктів у файл

У вікні, яке відкриється, побачимо всі доступні захоплені об'єкти (рис.4.22). Залишилось лише експортувати їх у файлову систему. Можемо зберігати як зображення, так і музику.

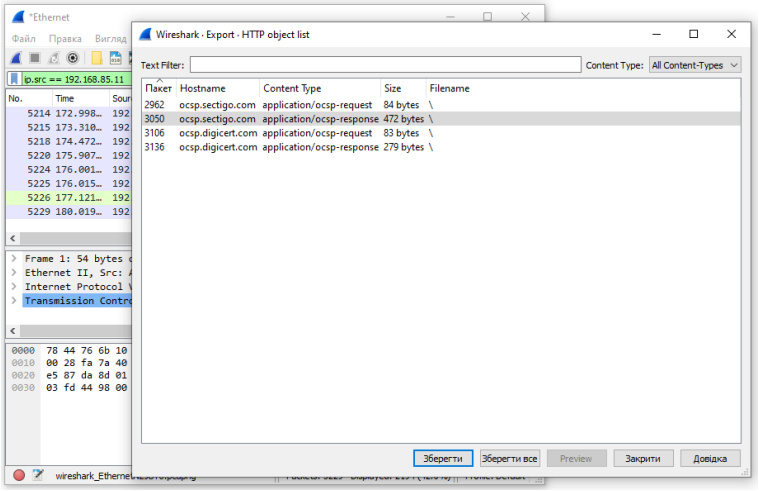


Рисунок 4.22 – Вікно із захопленими об'єктами

Далі можемо виконати аналіз мережевого трафіка Wireshark, або відразу відкрити отриманий файл іншою програмою, наприклад, плеєром.

### 4.4. Завдання

1. Встановити Wireshark.

3. Запустити Wireshark в режимі захоплення трафіка, який проходить через інтерфейс, підключений до локальної мережі.

4. Виконати емуляцію мережевої активності тривалістю 10-15 хвилин з різних домашніх вузлів. Для цього виконати деякі з наведених дій:

- Відкрити сайт http://.....
- Підключитися до серверу ftp.
- Підключитися до серверу mail.
- Підключитися до серверу ssh.
- Виконати пінг будь-яких вузлів.
- Підключитися до одного з доступних мережевих дисків Windows (якщо є в мережі).
- Виконати інші дії, які вимагають мережевого підключення.

5. Зупинити захоплення, зберегти pcap-файл.

6. Виконати всі пункти, розглянуті вище в даній лабораторній роботі.

### 4.5. Зміст звіту

1. Скріншоти виконання завдання.

2. Відповіді на контрольні питання до роботи.

3. Короткі висновки.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 4.6. Контрольні питання

1. Як відфільтрувати пакети, які мають більшу певного значення мережеву відстань?

2. Як відфільтрувати пакети певного відправника?

3. Як експортувати захоплені пакети у файл заданого типу?

## Лабораторна робота 5
## ЗАХОПЛЕННЯ І РОЗШИФРОВКА БЕЗДРОТОВОГО ТРАФІКУ

**Мета роботи:** Ознайомлення з методами захоплення трафіку зловмисниками в бездротовій мережі.

**Зміст.** В роботі розглядаються налаштування параметрів та використання для захоплення і аналізу бездротового трафіку за допомогою Wireshark на Raspberry Pi.

### 5.1. Теоретичні відомості

Для даної лабораторної роботи необхідно мати встановлений Kali Linux на мікрокомп'ютері RPi. Використання Wireshark на Windows, особливо для захоплення Wi-Fi трафіка, не рекомендується: по-перше, треба встановлювати додаткові драйвери – WinPcap, але головна проблема в тому, що режим моніторингу мережевого адаптера в Windows працює не завжди коректно.

Крім того, не кожен адаптер Wi-Fi здатний працювати в режимі моніторингу, а адаптер RPi працює в такому режимі без проблем.

В процесі використання Wireshark нас, в першу чергу, буде цікавити трафік, який передається через бездротові мережі. Режим моніторингу Wi-Fi адаптера дозволить бачити весь трафік, а не лише призначений нашій мережевій картці.

Вмикаємо:

```
airmon-ng start wlan0
```

Тепер можна запускати Wireshark: це можно зробити із вкладки "Додатки", а можна з терміналу:

```
sudo wireshark
```

Відразу після запуску, буде запропоновано вибрати мережевий інтерфейс для запуску захоплення (рис.5.1). Можемо це зробити, або, якщо хочемо

працювати з раніше захопленим трафіком, збереженим у файлі, можемо натиснути File -> Open і вибрати необхідний файл.

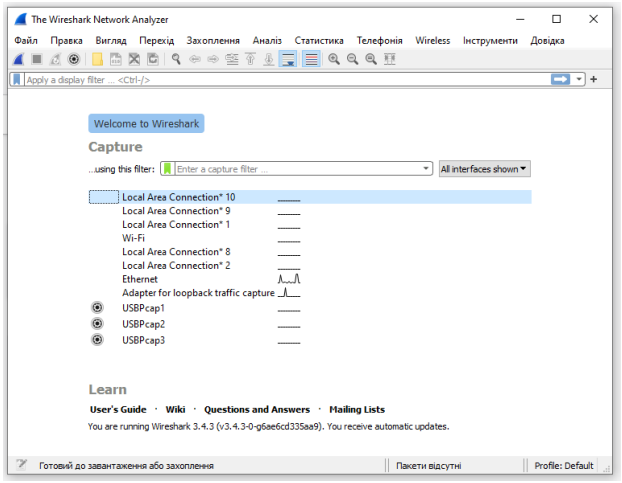


Рисунок 5.1 – Вибір інтерфейсу для захоплення трафіка

Обираємо той мережевий інтерфейс, який нас цікавить, наприклад, wlan0mon і двічі клацаємо на ньому. Перед нами з’явиться головне вікно програми, з яким будемо працювати далі, і відразу почнеться захоплення даних, поки не натиснемо «Стоп» (червоний квадратик ліворуч у верхньому куту).

**Налаштування Wireshark**

Загальні параметри самого додатка Wireshark знаходяться у вкладці Edit -> Preferences. Їх можна налаштувати в залежності від своїх потреб або бажань. Пізніше ми повернемося на цю вкладку, щоб дещо змінити, а зараз просто коротко розглянемо, які тут є вкладки (рис.5.2).

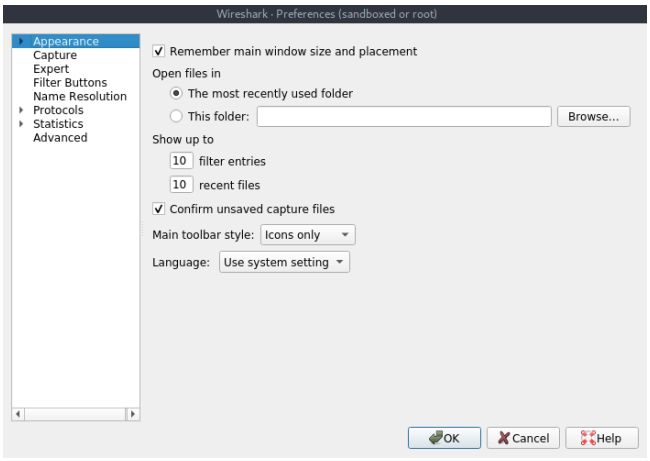


Рисунок 5.2 – Загальні параметри Wireshark

- **Appearance (Зовнішній вигляд)**. Тут визначається зовнішній вигляд програми: розміщення вікон, основних панелей, полоси прокрутки і стобців в панелі Packet List, використані шрифти, а також кольори фону і шрифтів.
- **Capture (Захоплення**). На цій вкладці можна налаштувати порядок захоплення пакетів, в тому числі стандартний інтерфейс для захоплення, параметри оновлення панелі Packet List в реальному часі.
- **Filter Expressions (Вирази фільтрування)**. Тут можна створювати фільтри мережевого трафіка і керувати ними.
- **Name Resolution (Перетворення імен)**. На цій вкладці активуються засоби Wireshark, які дозволяють пертворювати адреси в імена, більш зручні для розуміння, в тому числі адреси канального, мережевого і транспортного рівнів, а також можна вказувати максимальну кількість паралельних запитів на перетворення імен.
- **Protocols (Протоколи**). Тут знаходяться параметри, які впливають на захоплення і відображення різних пакетів, які Wireshark може декодувати. Хоча налаштовувати параметри можна не для всіх протоколів, але рідко виникає необхідність щось тут змінювати.
- **Statistics (Статистика)**. Назва вкладки говорить, що тут знаходяться параметри відображення і збереження статистики.
- **Advanced (Додаткові)**. Параметри, які не увійшли в жодну з перерахованих категорій, або призначені для більш тонкого наналаштування функціоналу.

Ми плануємо працювати з трафіком бездротових мереж, тому є сенс для більшої зручності трохи змінити робочі області Wireshark, а саме: додати кілька нових стовбців, інформація з яких допоможе значно заощадити час.

Перейдемо в Edit -> Preferences і в вкладці Appearance виберемо вкладку Columns, тут натискаємо на плюс і додаємо наступні стовбці з такими параметрами (рис.5.3):

- Title: Channel, Type: Custom, Fields: wlan_radio.channel – буде показувати нам канал в якому захоплений пакет;
- Title: Signal Strength, Type: Custom, Fields: wlan_radio.signal_dbm – покаже потужність сигналу в каналі в момент захоплення пакета.

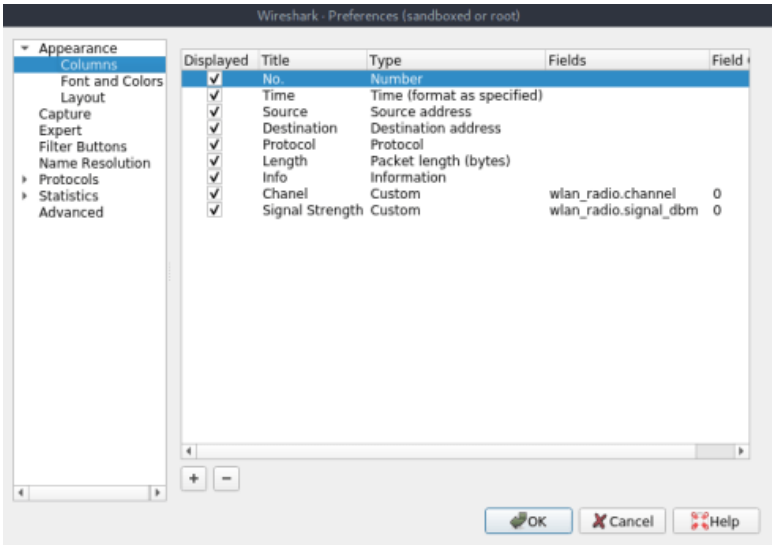


Рисунок 5.3 – Зміна робочих областей Wireshark

**Розшифровка бездротового трафіка**

Коли ми запустили захоплення трафіка на інтефейсі wlan0mon, то звернули увагу, що в Packet List все в такому вигляді, що з корисної інформації там можна знайти лише назву мережі (SSID) і MAC-адреси. Насправді, інформація є, просто вона зашифрована. Чому так? Відповідь ми знаємо – трафік шифрується. Відповідно, його треба розшифровати, а для цього треба розуміти деякі процеси, які відбуваються в Wi-Fi мережах.

При передачі даних через Wi-Fi трафік, зазвичай, шифрується з використанням ключа PTK (Pairwise Transient Key). При цьому цей ключ динамічний, тобто створюється заново для кожного нового з’єднання, і, відповідно, трафік для кожного з’єднання в одній і тій же мережі зашифрований різними PTK. Якщо клієнт перепідключається, то і PTK також змінюється.

Щоб отримати цей PTK необхідно перехопити чотирьохетапне рукостискання, ну і знати пароль, ім’я (SSID) Wi-Fi мережі і канал, на якому вона працює. Будемо вважати, що пароль точки доступу ми знаємо, а SSID і канал

бачимо в Wireshark, питанням залишається лише PTK. Відповідно, треба перехопити рукостискання і не будь-яке, а саме те, яке відбулось між «нашим» клієнтом і точкою доступу безпосередньо перед обміном необхідною нам інформацією. Оскільки наш адаптер уже в режимі моніторингу, то необхідні нам дані побачимо в Packet Details при натисканні на пакет з мережі, яка нас цікавить (рис.5.4):

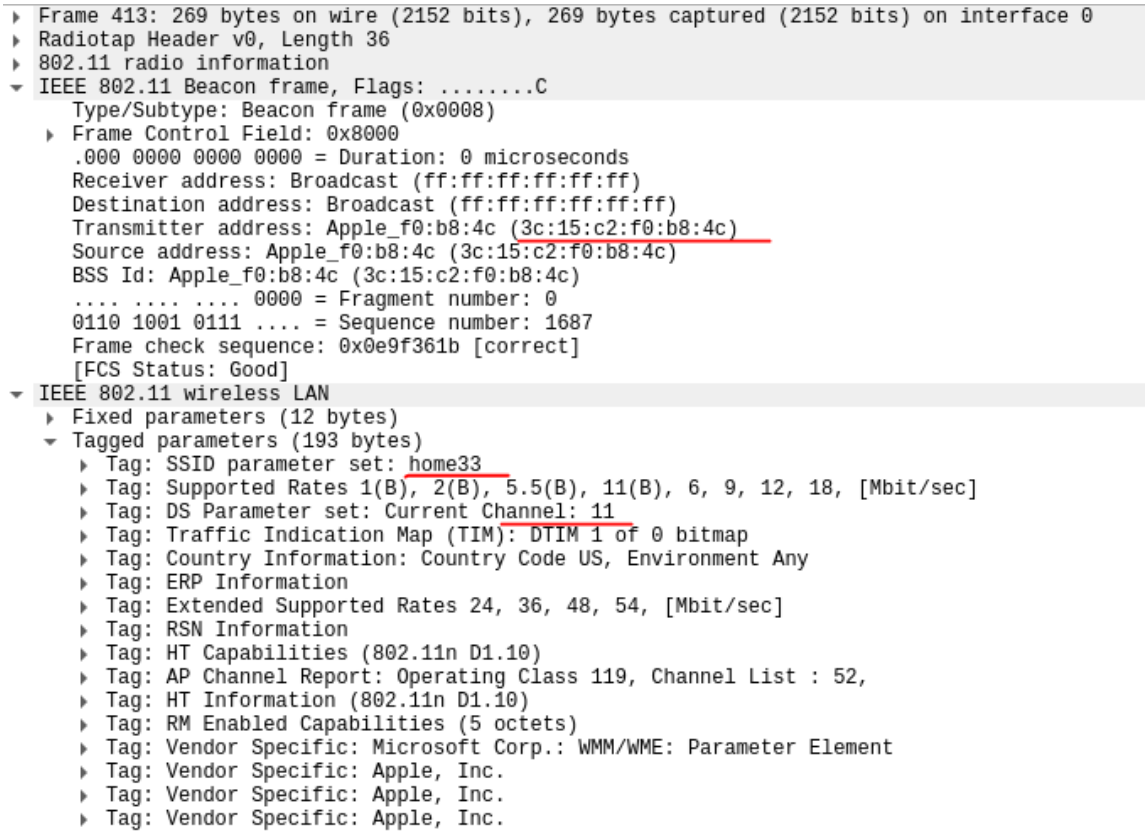


Рисунок 5.4 – Дані в Packet Details

Тепер відкриваємо термінал і запускаємо захоплення рукостискання:

```
sudo airodump-ng wlan0mon --channel 11 --write 123
```

Щоб не чекати повторних підключень, прискоримо процес, допомігши відключитися всім пристроям в мережі за допомогою команди деаутентифікації:

```
aireplay-ng --deauth 100 -a 3c:15:c2:f0:b8:4c wlan0mon
```

Через невеликий проміжок часу, коли пристрої знову спробують підключитися до точки доступу, у верхньому куту праворуч нашого терміналу побачимо напис, що з'явився, **WPA handshake** (рис.5.5).

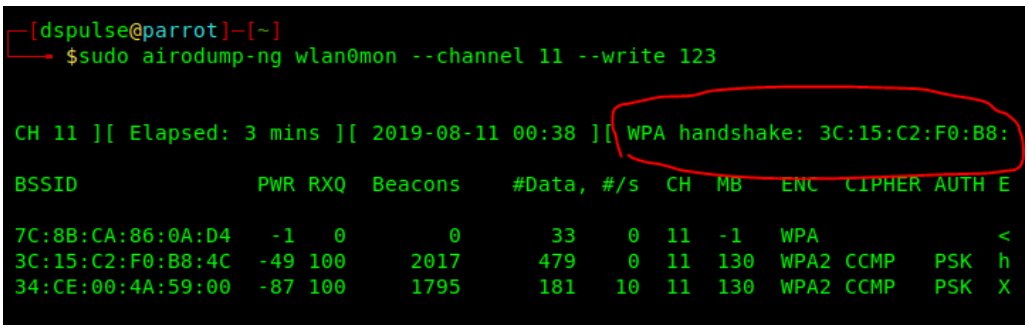


Рисунок 5.5 – Захоплення рукостискання

Це означає, що ми отримали рукостискання. Повернемося в Wireshark.

В Filter Toolbar треба написати (рис.5.6):

`eapol`

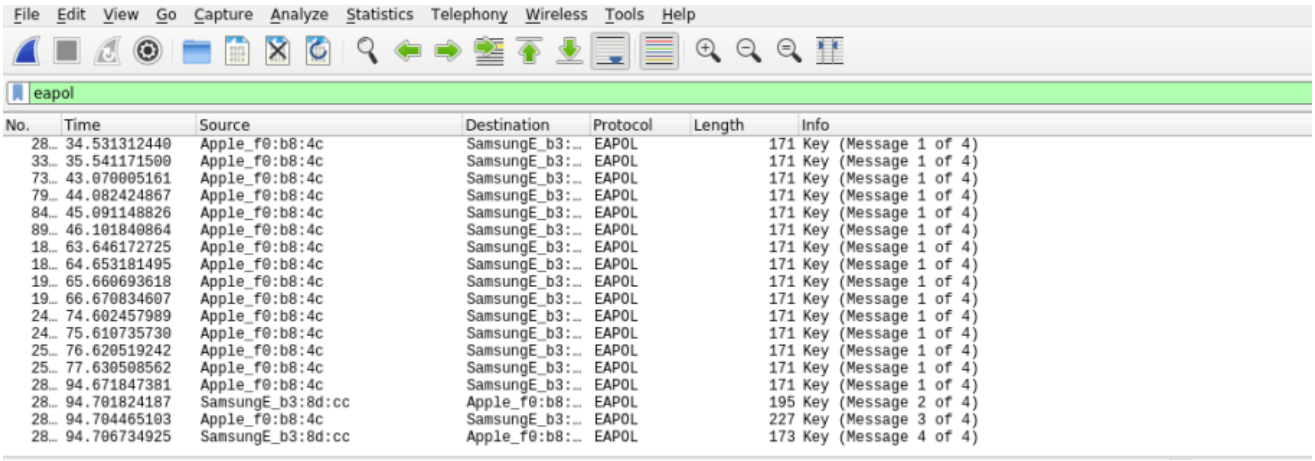


Рисунок 5.6 – Виконання `eapol`

Це необхідно, щоб переконатися, що рукостискання дійсно захоплене – якщо це так, то можемо продовжити. Теперь дещо змінимо налаштування протоколів. Переходимо Edit -> Preferences і вибираємо вкладку Protocols, в якій треба знайти IEEE 802.11 і поставити галочку Enable decryption (рис.5.7), а потім натиснути Edit.

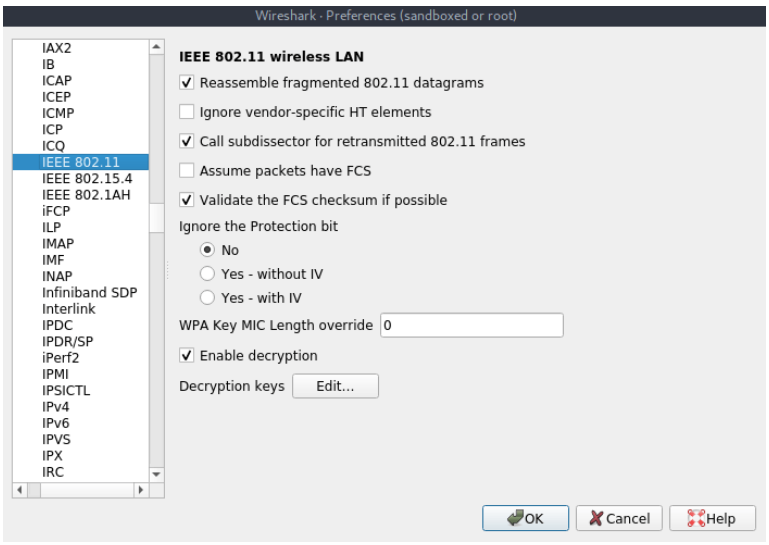


Рисунок 5.7 – Налаштування протоколу IEEE 802.11

У вікні, яке з’явиться, натискаємо “+” і в Key type треба вибрати wpa-pwd, а там, де Key, треба ввести через двокрапку “пароль:ім’я мережі” (рис.5.8). Натискаємо ОК і зберігаємо зміни в налаштуваннях протоколу.

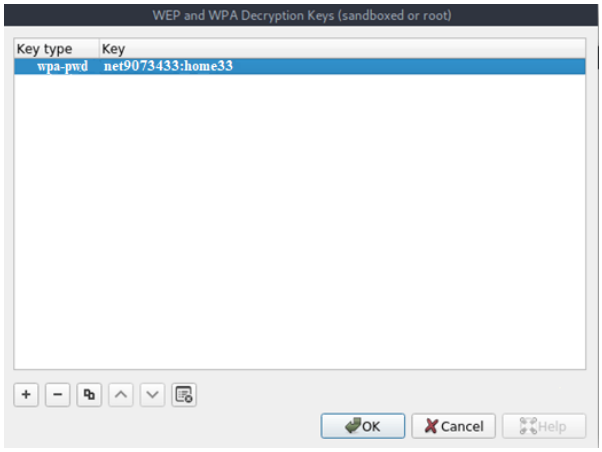


Рисунок 5.8 – Вікно зміни протоколу

Відразу після цього захоплений трафік розшифрується і матиме більш зрозумілий вигляд.

**Аналіз трафіку**

У більшості випадків (майже завжди) аналіз перехопленого трафіку відбувається в оффлайн-режимі тобто спочатку захоплюємо трафік, зберігаємо його в файлі захоплення (File -> Save), потім знову захоплюємо і знову зберігаємо, далі об'єднуємо всі файли захоплення в один (File -> Merge) і лише

потім аналізуємо весь трафік разом. До речі, ці функції, зі збереженням і об'єднанням, будемо використовувати досить часто.

Виконувати збереження необхідне тому, що отримати з першого разу відомості, які нам потрібні, вдається далеко не завжди, та й якщо аналізувати трафік в онлайн – обов’язково щось пропустимо. А об'єднання треба тому, що коли у нас є кілька файлів захоплення – розбирати їх кожен окремо – дуже трудомістко. Файли захоплення містять величезну кількість рядків і, щоб швидко в них орієнтуватися, треба навчиться робити кілька речей:

1. **Пошук за пакетами**. При натисканні комбінації клавіш Ctrl+F відкривається панель пошуку. Там є кілька варіантів пошуку, але на даному етапі досить буде використовувати Display filter, який дозволяє створити фільтр, щоб знайти тільки ті пакети, які відповідають заданому в ньому виразу і String – здійснює пошук в рядках за вказаними символами (пошук вперед Ctrl+N, пошук назад Ctrl+B).

2. **Відзначення пакетів**. Дуже часто буває, що був знайдений необхідний пакет, але треба буде повернутися до нього пізніше. Для цього цей пакет можна відзначити, натиснувши правою кнопкою на потрібний пакет в Packet List і вибрати Mark Packet або натиснути Ctrl+M. Відзначати можна скільки завгодно пакетів, а щоб переміщатися між відзначеними пакетами використовуються комбінації клавіш Shift+Ctrl+N – наступний і Shift+Ctrl+B – попередній.

3. **Фільтри**. Ми розглядали на попередньому занятті деякі приклади застосування фільтрів, тому не будемо на цьому зупинятися.

**Практика аналізу трафіку бездротових мереж**

Вже згадувалось, що для збору трафіка в бездротовій мережі потрібно використовувати режим моніторингу мережевого адаптера – в цьому режимі буде працювати наш мережевий адаптер. Хоча, для розуміння ситуації в цілому, непогано було б знати, в яких режимах можуть працювати мережеві адаптери:

• **Керований режим (Managed mode)**. Застосовується в тому випадку, якщо клієнт бездротової мережі підключається безпосередньо до точки

бездротового доступу. У подібних випадках програмний драйвер, зв'язаний з адаптером бездротового зв'язку, використовує точку бездротового доступу для управління всім процесом обміну даними в бездротовій мережі.

• **Режим прямого підключення (Ad-hoc mode)**. Застосовується в тому випадку, якщо організована бездротова мережа, в якій пристрої підключаються безпосередньо один до одного. В цьому режимі два клієнта бездротової мережі, яким потрібно обмінюватися даними один з одним, поділяють обов'язки, які зазвичай покладаються на точку бездротового доступу.

• **Ведучий режим (Master mode)**. Деякі адаптери бездротового зв'язку підтримують також провідний режим. В цьому режимі адаптеру бездротового зв'язку дозволяється працювати разом зі спеціальним програмним драйвером, щоб комп'ютер, на якому встановлений цей адаптер, діяв як точка бездротового доступу для інших пристроїв.

• **Режим моніторингу, який ще називають режимом контролю (Monitor mode)**. Це той самий режим, який будемо використовувати для захоплення і аналізу пакетів. Дозволяє прослуховувати пакети, поширювані в ефірі. Для повноцінного захоплення і аналізу пакетів адаптер разом з програмним драйвером повинен підтримувати режим поточного контролю, так званий режим RFMON, тобто режим радіочастотного контролю.

Схематично принцип дії режимів можно зобразити так (рис.5.9):

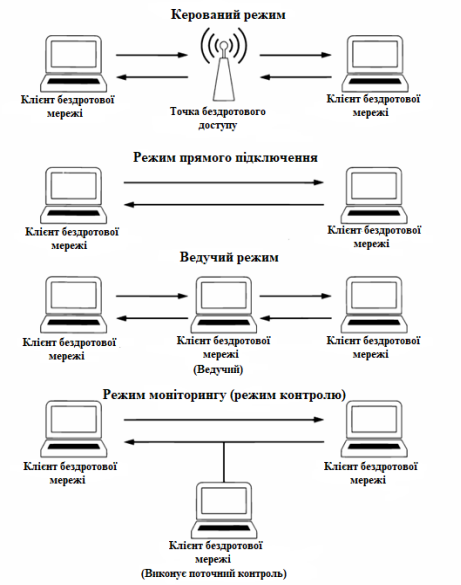


Рисунок 5.9 – Режими роботи бездротових адаптерів

В бездротовій мережі весь мережевий трафік, сформований бездротовими кдієнтами, співіснує в загальних каналах. Це означає, що захоплені пакети в будь-якому одному каналі можуть нести інформацію від кількох клієнтів, відповідно, щоб знайти цікаву для нас інформацію треба навчитися відфільтровувати зайве. Для цього давайте розглянемо фільтри, які найчастіше використовують при аналізі бездротових мереж:

**Фільтрація за ідентифікатором**: Wireshark збирає дані всіх мереж, які знаходяться в радіусі дії мережевого адаптера, тому логічно спочатку відфільтрувати трафік конкретної мережі, яка нас цікавить. Зробити це можна за SSID бездротової мережі:

`wlan.ssid == ім'я_мережі`

Хоча більш коректно буде виконати фільтрацію за `bssid`. BSSID – це ідентифікатор базового набору послуг – він присвоюється кожній точці та ідентифікує її. При цьому даний ідентифікатор надсилається в кожному бездротовому пакеті управління і пакеті даних з передавальної точки доступу. BSSID записується в заголовок пакета і це буде MAC-адреса нашої точки доступу. Побачивши його в заголовку (рис.5.10), можемо створити фільтр, щоб бачити трафік, який проходить лише через необхідну точку доступа (рис.5.11):

`wlan.bssid == 3c:15:c2:f0:b8:4c`

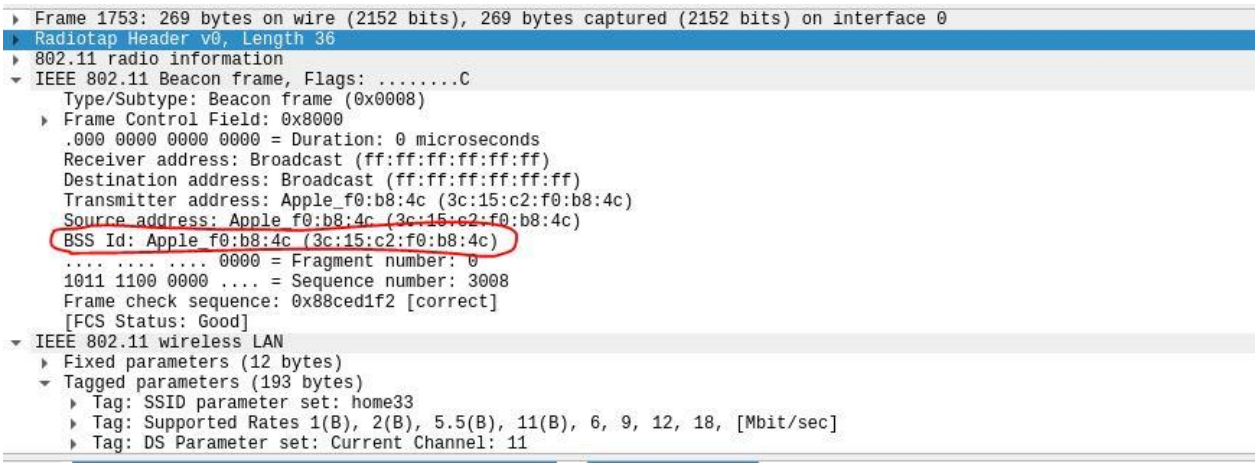


Рисунок 5.10 – BSSID в заголовку пакета

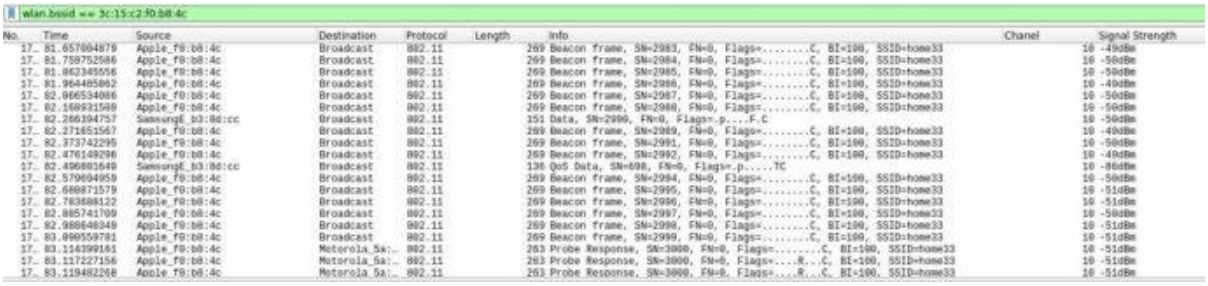


Рисунок 5.11 – Робота фільтра за BSSID

Не менш корисним буде в деяких ситуаціях фільтрування трафіку за використовуваним каналом зв’язку:

```
wlan_radio.channel == 10
```

Використовуємо фільтр (рис.5.12):

```
arp
```

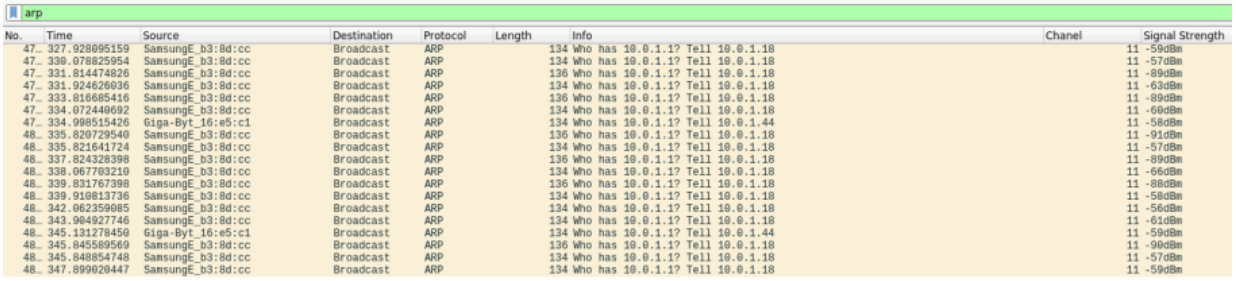


Рисунок 5.12 – Використання фільтра `arp`

І тепер можемо побачити трафік, який передається через протокол ARP. Це дає можливість зрозуміти, які пристрої в даний момент підключені до локальної мережі, побачити їх MAC і IP адреси.

Також досить часто використовуються такі фільтри як:

```
dns
```

Цей фільтр покаже відправлені dns-запити і можна буде дізнатися, які сайти відвідував користувач і якими онлайн-ресурсами користувався.

Фільтр

```
ip.addr == 192.168.183.129
```

покаже трафік, зв’язаний з конкретним IP (де він був отримувачем або відправником).

`tcp`

покаже tcp трафік. За таким же принципом можна відфільтрувати трафік за будь-яким іншим протоколом, наприклад, udp або icmp.

Якщо бачимо, що з’єднання з сайтом не захищене, тобто здійснюється через протокол http, це відкриває широкі можливості зловмиснику, бо він зможе побачити дані, які передаються, в тому числі дані авторизації і дані форм, файли, що завантажуються чи відкриваються, передані і встановлені cookie (табл.5.1).

Таблиця 5.1 – Фільтри для захоплення даних через протокол http

| | |
|---|---|
| http | Відфільтрувати http-трафік |
| http.host == “*адреса*” | Показати запити до певного сайту |
| http.cookie | http-запити, в яких передавались cookie |
| http.set_cookie | Запити, в яких були встановлені cookie в браузер |
| http.content_type contains “image” | Пошук будь-яких переданих зображень, можна конкретизувати, змінивши “image” на “jpeg” або інші розширення. |
| http.authorization | Пошук запитів авторизації |
| http.request.uri contains “zip” | Пошук файлів певного типу (zip змінити на необхідне) |

До речі, нагадуємо, що для збереження будь-якого знайденного файла, треба натиснути на ньому правою кнопкою миші у вікні Packet Details і вибрати Export Packet Bytes та вказати місце, в якому його треба зберегти.

**Фільтрація захопленого POST трафіка**

Багато користувачів не знають, що заповнюючи логін і пароль при реєстрації або авторизації на закритому Інтернет-ресурсі і натискаючи ENTER, ці дані можуть бути легко перехоплені. Дуже часто вони передаються через мережу в незахищеному вигляді. Тому, якщо сайт, на якому ми намагаємося

авторизуватися, використовує HTTP протокол, то дуже просто виконати захоплення цього трафіку, проаналізувати його за допомогою Wireshark і далі за допомогою спеціальних фільтрів та програм знайти і розшифрувати пароль.

Найкраще місце для перехоплення паролів – ядро мережі, де проходить трафік всіх користувачів до закритих ресурсів (наприклад, пошта) або перед маршрутизатором для виходу в Інтернет, при реєстраціях на зовнішніх ресурсах тощо.

Для перевірки запускаємо захоплення трафіку. Відкриваємо браузер і намагаємося авторизуватися на будь-якому ресурсі за допомогою логіна і пароля. Після завершення процесу авторизації і відкриття сайту зупиняємо захоплення трафіку в Wireshark.

Далі відкриваємо аналізатор протоколів і бачимо велику кількість пакетів. Нас цікавлять конкретні пакети, які містять POST-дані, що формуються на нашій локальній машині при заповненні форми на екрані і відправляються на віддалений сервер при натисканні кнопки «Вхід» або «Авторизація» в браузері. Вводимо у вікні спеціальний фільтр для відображення захоплених пакетів: `http.request.method == "POST"`. Тепер замість тисячі пакетів бачимо лише один з даними, які шукаємо (рис.5.13).

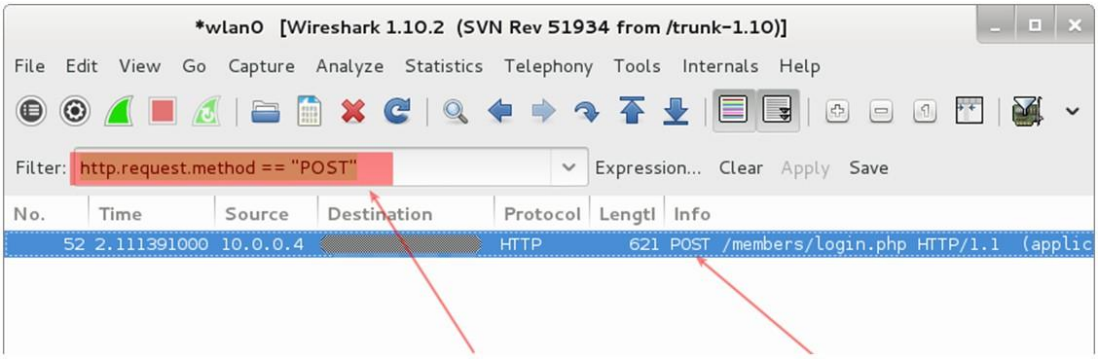


Рисунок 5.13 – Застосування фільтра `http.request.method == "POST"`

**Знаходимо логін і пароль користувача**

Клацаємо правою кнопкою миші на знайденому пакеті і вибираємо з меню пункт **Follow TCP Steam** (рис.5.14).

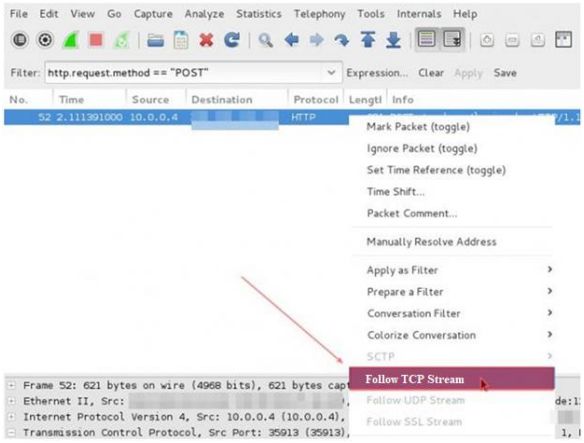


Рисунок 5.14 – Перехід до відновлення тексту

Після цього в новому вікні з’явиться текст, який в коді відновлює вміст сторінки. Знайдемо поля «password» і «user», які відповідають паролю та імені користувача. В деяких випадках обидва поля будуть навіть не зашифровані, але якщо ми спробуємо захопити трафік при звертанні до відомих ресурсів типу Gmail.com, fb.com тощо, то пароль буде хешований:

```
HTTP/1.1 302 Found
Date: Mon, 10 Nov 2014 23:52:21 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Set-Cookie: non=non; expires=Thu, 07-Nov-2024 23:52:21 GMT; path=/
Set-Cookie: password=e4b7c855be6e3d4307b8d6ba4cd4ab91;
expires=Thu, 07-Nov-2024 23:52:21 GMT; path=/
Set-Cookie: scifuser=networkguru; expires=Thu, 07-Nov-2024
23:52:21 GMT; path=/
Location: loggedin.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

Таким чином, в нашому випадку:

Ім’я користувача: networkguru

Пароль (хеш): e4b7c855be6e3d4307b8d6ba4cd4ab91

**Визначення типу хешування і розшифровка паролю**

Заходимо, наприклад, на сайт http://www.onlinehashcrack.com/hash-identification.php#res і вводимо наш пароль у вікно для ідентифікації. Видається список протоколів хешування в порядку приорітету:

- MD5
- NTLM
- MD4
- LM

На даному этапі можна скористатися утилітою `hashcat`:

```
~# hashcat -m 0 -a 0 /root/wireshark-hash.lf /root/rockyou.txt
```

На виході можемо отримати розшифроваий пароль: *simplepassword*.

Таким чином, за допомогою Wireshark можна не лише вирішувати проблеми в роботі додатків та сервісів, але і наглядно побачити, як зловмисники здійснюють перехоплення паролів, які користувачі вводять у веб-формах. Як бачите, не так вже й складно це зробити.

Щоб дізнатися паролі до поштових скриньок користувачів, використовують нескладні фільтри для відображення:

- Протокол POP і фільтр виглядають наступним чином: `pop.request.command == "USER" || pop.request.command == "PASS"`
- Протокол IMAP і фільтр буде: `imap.request contains "login"`
- Протокол SMTP і вимагання вводу – наступний фільтр: `smtp.req.command == "AUTH"`

Рекомендується виконати наведені вправи щоб переконатися в надійності тих поштових сервісів, якими ми користуємось. Пам’ятаємо, що найкраще організувати захист тоді, коли знаєш, яким шляхом користуються зловмисники при доступі до інформації.

**Дії зловмисника, якщо трафік зашифрований і використовується HTTPS**

Є кілька варіантів.

*Варіант 1*. Підключається в розрив з'єднання між користувачем і сервером та спробує захопити трафік в момент встановлення з'єднання (SSL Handshake). В момент встановлення з'єднання зловмисник може перехопити сеансовий ключ.

*Варіант 2*. Зловмисник може розшифрувати трафік HTTPS, використовуючи файл журнала сеансових ключів, які записує Firefox або Chrome. Для цього браузер повинен бути налаштований для запису цих ключів шифрування в файл журналу і зловмисник повинен отримати доступ до файла журналу. Власне, виконується крадіжка файла з ключем сесії з жорсткого диска іншого користувача, що є протизаконним. Далі захоплюється трафік і застосовується отриманий ключ для його розшифровки.

Після отримання ключів за вказаними варіантами в Wireshark прописується:

1. Перехід меню Edit – Preferences – Protocols – SSL.
2. Ставиться прапор «Reassemble SSL records spanning multiple TCP segments» (рис.5.15).
3. «RSA keys list» і натискається Edit.
4. Вводяться дані в усі поля і прописується шлях до файла з ключем.

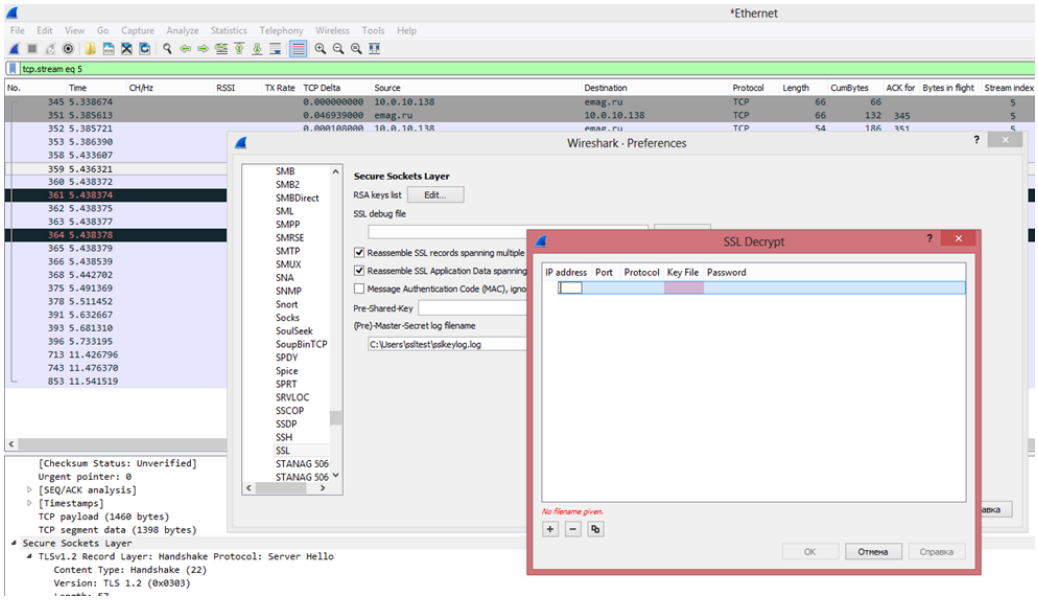


Рисунок 5.15 – Розшифровування паролю

Wireshark може розшифровувати пакети, які зашифровані з використанням алгоритму RSA. Якщо ж використовуються алгоритми DHE/ECDHE, FS, ECC, сніфер буде безпорадним, тому рекомендується використовувати ці алгоритми при роботі в мережі.

### 5.2. Завдання

1. Виконати вправи, наведені в даній лабораторній роботі. При отриманні повідомлення про помилку виконання або при успішному виконанні, зробити скріншоти для звіту.

### 5.3. Зміст звіту

1. Скріншоти виконання роботи.

2. Висновки за результатами виконання роботи.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 5.4. Контрольні питання

1. Який алгоритм шифрування доступний для розшифрування за допомогою Wireshark?

2. Які існують режими роботи бездротових адаптерів?

3. Чому для захоплення трафіку використовують режим моніторингу?

4. Як прискорити процес перепідключення пристроїв в мережі?

# Лабораторна робота 6
# ІНВЕНТАРИЗАЦІЯ МЕРЕЖЕВИХ РЕСУРСІВ З ВИКОРИСТАННЯМ УТИЛІТИ NMAP

**Мета роботи:** Вивчити основні прийоми роботи з програмою Nmap.

**Зміст.** Розглядається встановлення Nmap, приклади виконання окремих команд: виявлення номерів версій, сканування, перевірка вразливості Diffie-Hellman, виявлення захисту брандмауером, виявлення відкритих портів тощо.

## 6.1. Загальні відомості

Nmap – це утиліта з відкритим вихідним кодом для дослідження мережі та перевірки безпеки. Вона була розроблена для швидкого сканування великих мереж, хоча прекрасно справляється і з одиничними цілями.

Nmap використовує безліч різних методів сканування, таких як UDP, TCP (connect), TCP SYN (напіввідкрите), FTP proxy (прорив через ftp), Reverse-ident, ICMP (ping), FIN, ACK, Xmas tree, SYN та NULL-сканування.

Nmap також підтримує великий набір додаткових можливостей, а саме: «невидиме» сканування, динамічне обчислення часу затримки та повтор передачі пакетів, паралельне сканування, визначення неактивних хостів методом паралельного ping-опитування, сканування з використанням хибних хостів, визначення наявності пакетних фільтрів (без використання portmapper) RPC-сканування, сканування з використанням IP-фрагментації, а також довільне задання IP-адрес та номерів портів сканованих мереж.

У той час, як Nmap зазвичай використовується для перевірки безпеки, багато системних адміністраторів знаходять її корисною для звичайних завдань, таких як контролювання структури мережі, управління розкладами запуску служб і облік часу роботи хоста чи служби.

Вихідні дані Nmap – це список просканованих цілей з додатковою інформацією щодо кожної з них в залежності від заданих опцій. Ключовою інформацією є «таблиця важливих портів». Ця таблиця містить номер порту,

протокол, ім'я служби і стан. Стан може мати значення **`open`** (відкритий), **`filtered`** (фільтрується), **`closed`** (закритий) або **`unfiltered`** (не фільтрується).

**`Відкрито`** означає, що додаток на цільовій машині готовий для встановлення з'єднання/прийняття пакетів на цей порт.

**`Фільтрується`** означає, що брандмауер, мережевий фільтр, або якась інша перешкода в мережі блокує порт, і Nmap не може встановити відкритий цей порт чи закритий.

**`Закриті`** порти не пов'язані ні з яким додатком, але можуть бути відкриті в будь-який момент.

**`Не фільтровані`** – це коли порти відповідають на запити Nmap, але Nmap не може визначити відкриті вони чи закриті. Nmap видає комбінації **`відкритий|фільтрується`** і **`закритий|фільтрується`**, коли не може визначити, який з цих двох станів описує порт. Ця таблиця також може надавати деталі про версії програмного забезпечення, якщо це було запрошено. Коли здійснюється сканування через IP-протокол (`-sO`), Nmap надає інформацію про підтримувані протоколи, а не про відкриті порти.

На додаток до таблиці важливих портів Nmap може надавати подальшу інформацію про цілі: перетворені DNS-імена, припущення про використовувану операційну систему, типи пристроїв і MAC-адреси.

Nmap можна встановити на Windows, Linux, OSX.

## 6.2. Встановлення Nmap

Переходимо за посиланням завантаження nmap і завантажуємо останню стабільну версію (рис.6.1):

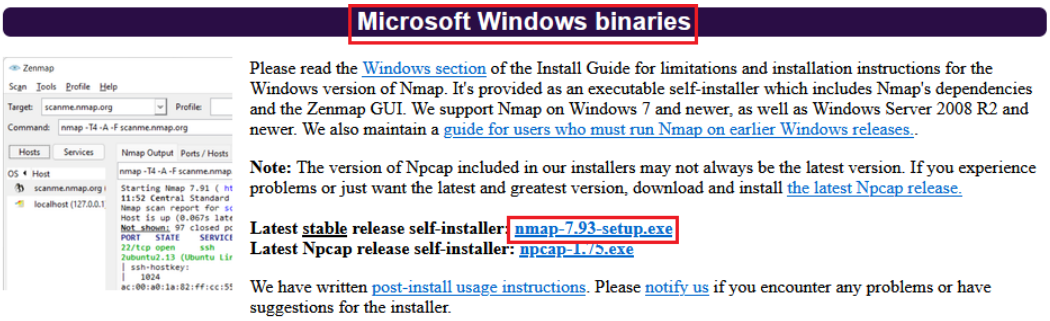


Рисунок 6.1 – Сторінка завантаження nmap

або, використаємо пряме посилання, щоб завантажити Nmap.

Переходимо в папку, куди завантажений файл.

Клацнемо правою кнопкой миші на EXE-файлі і виберемо «Run as administrator», щоб запустити від імені адміністратора:

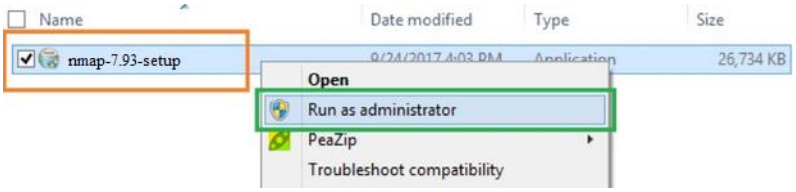


Рисунок 6.2 – Встановлення з правами адміністратора

Коли почнеться процес встановлення, приймаємо ліцензійну угоду:

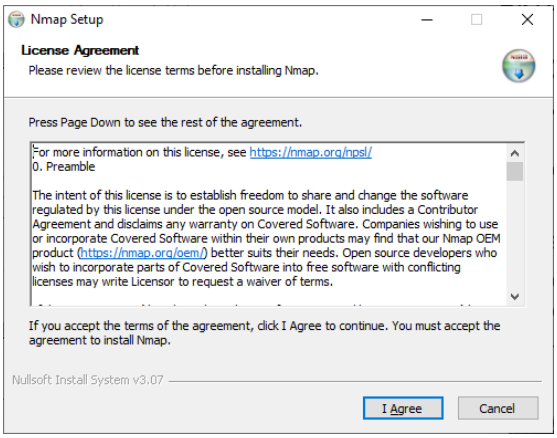


Рисунок 6.3 – Ліцензійна угода

Можемо вибрати компоненти для встановлення, але було б непогано встановити їх всі:

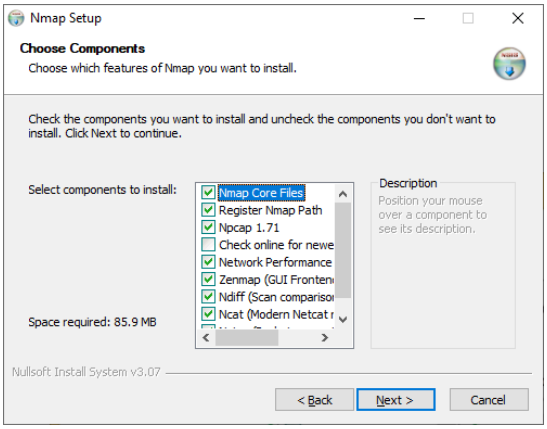


Рисунок 6.4 – Вибір компонент для встановлення

За замовчуванням Nmap буде встановлений в папці `C:\Program Files (x86)\Nmap`, але при необхідності можемо змінити його шлях:

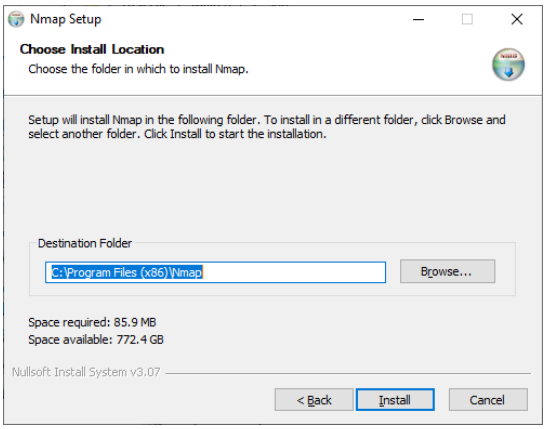


Рисунок 6.5 – Вибір шляху для встановлення програми

Почнеться встановлення Nmap і після його завершення, отримаємо підтвердження:

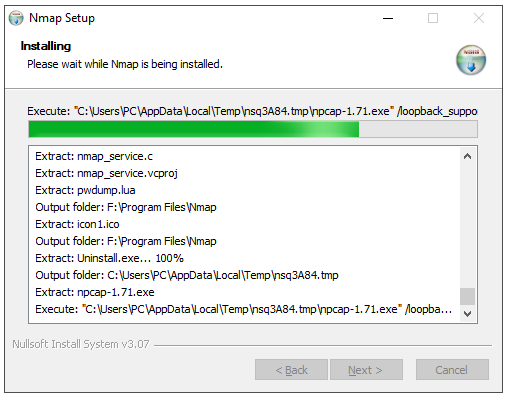


Рисунок 6.6 – Процес встановлення Nmap

Оскільки ми обрали серед компонент Npcap, то паралельно запускається встановлення і починається він з підтвердженням ліцензії (рис.6.7).

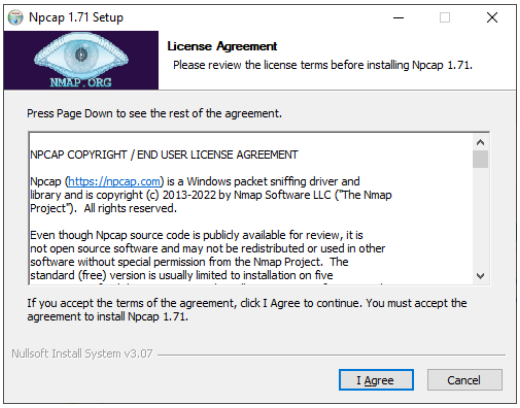


Рисунок 6.7 – Підтвердження ліцензії для Npcap

Задаємо необхідні параметри і завершуємо встановлення Npcap (рис.6.8)

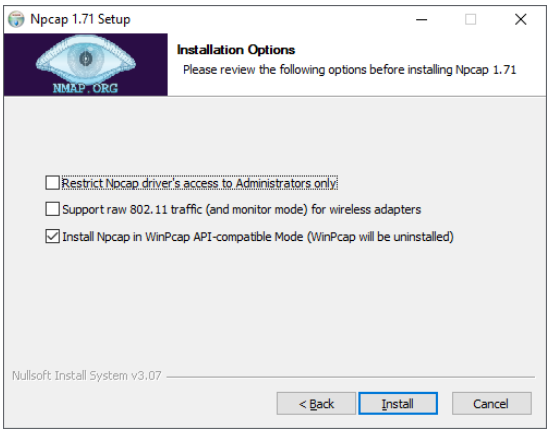


Рисунок 6.8 – Вибір параметрів Npcap

## 6.3. Приклади виконання деяких команд Nmap

### Виявлення номера версій

Одним із застосувань Nmap є відображення відбитку ОС і запущених служб:

```
nmap -sV $target
```

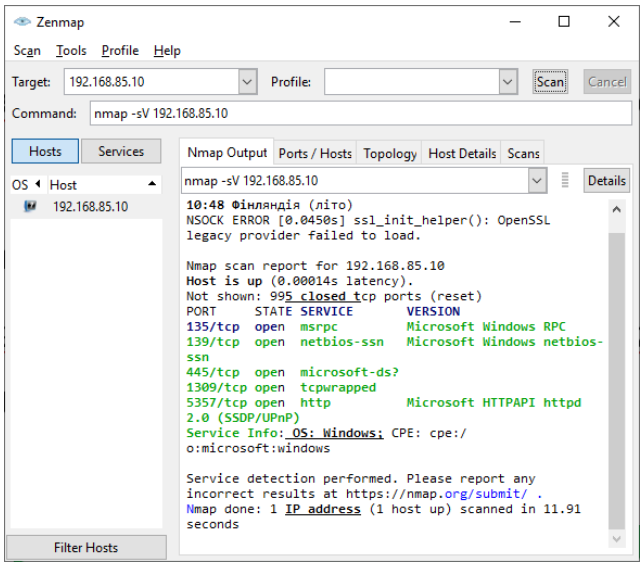


Рисунок 6.9 – Результат виконання команди

Аргумент команди `-sV` буде визначати версії і стан запущених служб.

### Сканування google.com

Виконаємо команду:

```
nmap -T4 -A -v google.com
```

Результат виконання показаний на рис.6.10.

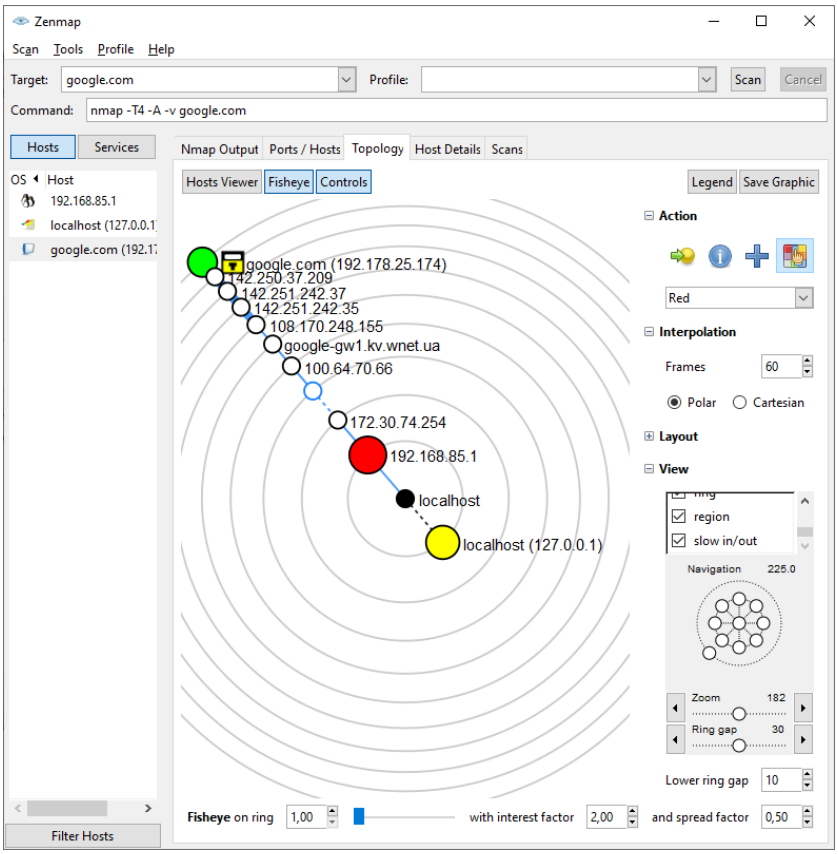


Рисунок 6.10 – Вкладка Topology результату сканування

## Перевірка вразливості Diffie-Hellman

Щоб виявити вразливість DH, можемо скористатися синтаксисом нижче:

```
nmap -p $port --script ssl-dh-params.nse $target
```

Результат виконання на рис.6.11.

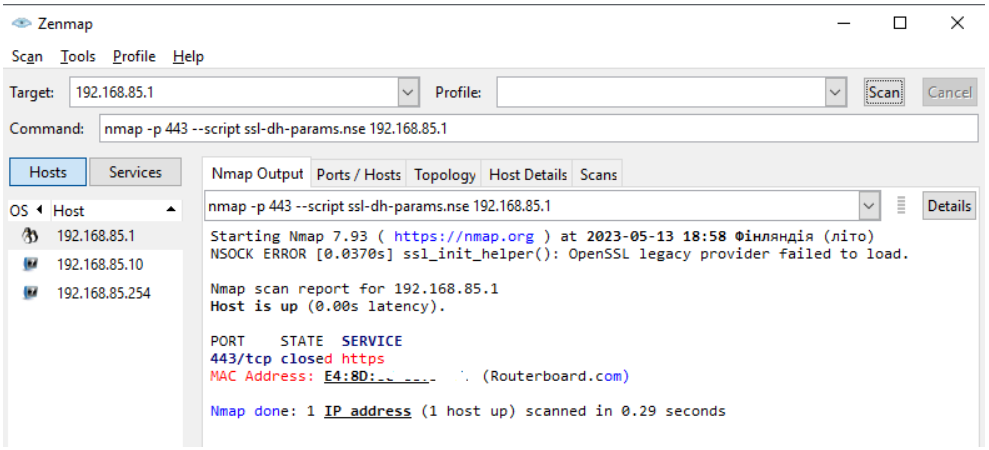


Рисунок 6.11 – Виявлення вразливості DH

## 6.4. Готові бібліотеки NSE

У Nmap вбудовано багато додаткових бібліотек, спеціально заточених під виконання різних мережевих перевірок. Всі вони знаходяться у папці `C:\Program Files (x86)\Nmap\nselib`.

Щоб не придумувати, наприклад, як виконати запит до DNS, розглянемо список бібліотек, що найчастіше використовуються:

• backdoor – найпростіша реалізація бекдору в системі;

• bin – робота з двійковими даними;

• datafiles – бібліотека для роботи із вбудованими файлами Nmap (наприклад, nmapprotocols – описом та номерами портів різних протоколів);

• default – мережні сценарії, які найчастіше використовуються;

• dns – робота з DNS;

• http – робота з HTTP;

• msrpc – бібліотека для дзвінків MSRPC;

• mysql – деякі операції з MySQL;

• netbios – робота з NetBIOS-трафіком;

• nmap – взаємодія з ядром сканера;

• packer – маніпулювання з пакетами у RAW-виді;

• proxy – робота з проксі;

• shortport – робота з портами;

• smb – робота з SMB-протоколом;

• sss1/ssh2 – робота з SSH.

## 6.5. Команди nmap, які часто використовуються зловмисниками

### Виявлення захисту брандмауером або фільтрації пакетів у хоста/мережі

Іноді потрібно дізнатися, на яких цільових машинах є фільтрація пакетів. Виявити брандмауер, що функціонує, можна за допомогою опції `-sA`.

Скануємо на наявність брандмауера за IP-адресою:

```
nmap -sA 192.168.1.254
```

Скануємо на наявність брандмауера на ім'я хоста:

```
nmap -sA server1.sedicomm.com
```

**Сканування хоста, захищеного брандмауером**

Також може виникнути потреба у скануванні захищеного хоста. Зробити це допоможе опція `-PN`, що вказує утиліті Nmap пропустити етап пінгу сканування.

Сканування хоста, захищеного брандмауером, за його IP-адресою:

```
nmap -PN 192.168.1.1
```

Сканування хоста, захищеного брандмауером, на ім'я хоста:

```
nmap -PN server1.sedicomm.com
```

**Виявлення всіх відкритих портів**

Однією з найкорисніших функцій Nmap є можливість шукати відкриті порти. Спеціально для цього призначено опцію `--open`.

Знаходимо відкриті порти за IP-адресою:

```
nmap --open 192.168.1.1
```

Знаходимо відкриті порти на ім'я хоста:

```
nmap --open server1.sedicomm.com
```

**Скануємо потайки портів TCP з TCP SYN**

Можна виконати напіввідкрите сканування (TCP SYN) без встановлення повноцінного підключення. Для цього скористаємося опцією `-sS`:

```
nmap -sS 192.168.1.1
```

Шукаємо за допомогою TCP-підключення порти, які найчастіше використовуються:

```
nmap -sT 192.168.1.1
```

Шукаємо за допомогою TCP ACK порти, які найчастіше використовуються:

```
nmap -sA 192.168.1.1
```

Шукаємо за допомогою сканування TCP Window порти, які найчастіше використовуються

```
nmap -sW 192.168.1.1
```

Шукаємо за допомогою сканування TCP Maimon порти, які найчастіше використовуються

```
nmap -sM 192.168.1.1
```

**Сканування брандмауера для виявлення вразливостей**

Є три типи сканування, які дають змогу отримувати відомості про налаштування безпеки віддалених машин.

**Скануємо TCP Null, що дозволяє «обдурити» брандмауер для отримання відповіді**

Перший варіант залишає заголовок прапора TCP рівним нулю (не встановлює бітів) за допомогою опції `-sN`:

```
nmap -sN 192.168.1.254
```

**Скануємо TCP FIN для перевірки брандмауер**

Другий варіант сканування встановлює лише біт TCP FIN за допомогою опції `-sF`:

```
nmap -sF 192.168.1.254
```

**Скануємо TCP Xmas для перевірки брандмауера**

Третій варіант за допомогою опції `-sX` встановлює прапори FIN, PSH та URG, підсвічуючи пакет:

```
nmap -sX 192.168.1.254
```

**Сканування брандмауера на наявність фрагментів пакетів**

Прапор `-f` змушує це сканування (включаючи перевірки ping) використовувати крихітні фрагментовані IP-пакети. Ідея полягає в тому, щоб розділити заголовок TCP на кілька пакетів для ускладнення фільтрації пакетів і, отже, ускладнити роботу систем виявлення вторгнень тощо:

```
nmap -f 192.168.1.1
nmap -f fw2.nixcraft.net.in
nmap -f 15 fw2.nixcraft.net.in

## Set your own offset size with the --mtu option ##
nmap --mtu 32 192.168.1.1
```

**Приховане сканування з обманками**

Існує можливість при скануванні приховати свою IP-адресу серед будь-якої кількості фіктивних адрес-обманок. Для цього призначено опцію `-D`:

```
nmap -n -D decoy-ip1,decoy-ip2,your-own-ip,decoy-ip3,decoy-ip4
remote-host-ip
nmap -n -D 192.168.1.5,10.5.1.2,172.1.2.4,3.4.2.1 192.168.1.5
```

**Спуфінг MAC-адрес**

Утиліта Nmap має у своєму арсеналі функцію спуфінгу MAC-адрес. Слід зазначити, що spoofing attack – це ситуація, при якій один пристрій/додаток маскується під інший, вказуючи його MAC-адресу.

Вказуємо підроблену MAC-адресу:

```
nmap --spoof-mac MAC-ADDRESS-HERE 192.168.1.1
```

Вказуємо додаткові опції спуфінгу:

```
nmap -v -sT -PN --spoof-mac MAC-ADDRESS-HERE 192.168.1.1
```

Використовуємо випадкову MAC-адресу

У цьому прикладі «0» означає, що Nmap вибере цілком випадкову MAC-адресу:

```
nmap -v -sT -PN --spoof-mac 0 192.168.1.1
```

## 6.6. Завдання

1. Перейти в командний рядок або скористатися графічним інтерфейсом.

2. Запустити nmap без вказівки параметрів, подивитися номер версії.

3. Виконати ідентифікацію вузлів в локальній мережі:

```
nmap -sP -n 192.168.x.1-254
```

4. Подивитися результати (рис.6.12:

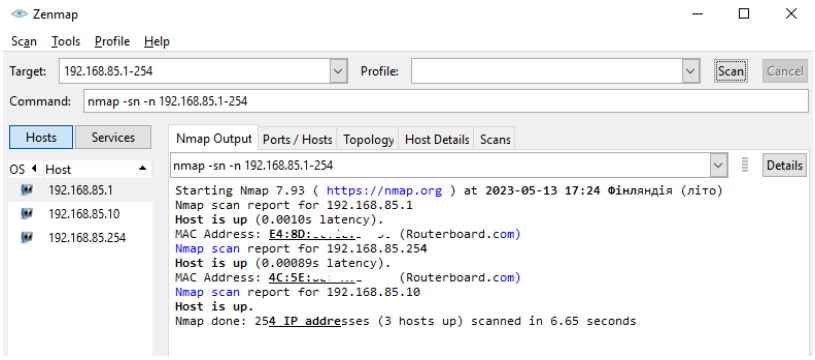


Рисунок 6.12 – Приклад ідентифікації вузлів

5. Повторити попередню команду, додавши опцію `-v` для отримання більше деталей в результатах:

```
nmap -sP -n -v 192.168.x.1-254
```

6. Переглянути результати, звернути увагу на використаний метод ідентифікації мережі:

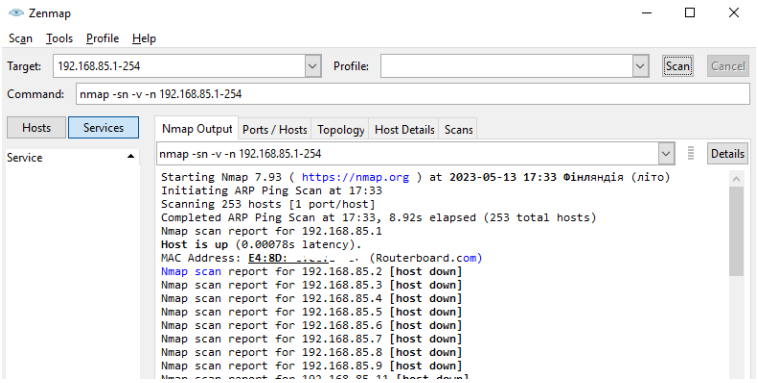


Рисунок 6.13 – Приклад більш детальної ідентифікації вузлів

7. Вибрати серед мережевих об’єктів один (наприклад, `192.168.x.1`) і виконати ідентифікацію відкритих TCP-портів:

```
nmap -sS -n 192.168.x.1
```

8. Переглянути результати. Серед переліку відкритих портів TCP вибрати один довільний, наприклад, 21, 25 або 80.

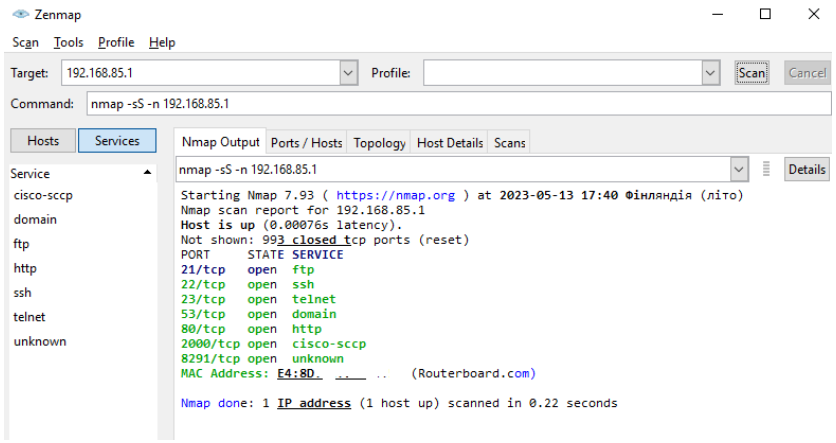


Рисунок 6.14 – Приклад ідентифікації відкритих TCP-портів

9. Виконати для вибраного порта ідентифікацію служби додатку:

```
nmap -A -p21 -n 192.168.x.200
```

10. Подивіться на результати (чи правильно ідентифікована служба?), зверніть також увагу на результати ідентифікації ОС:

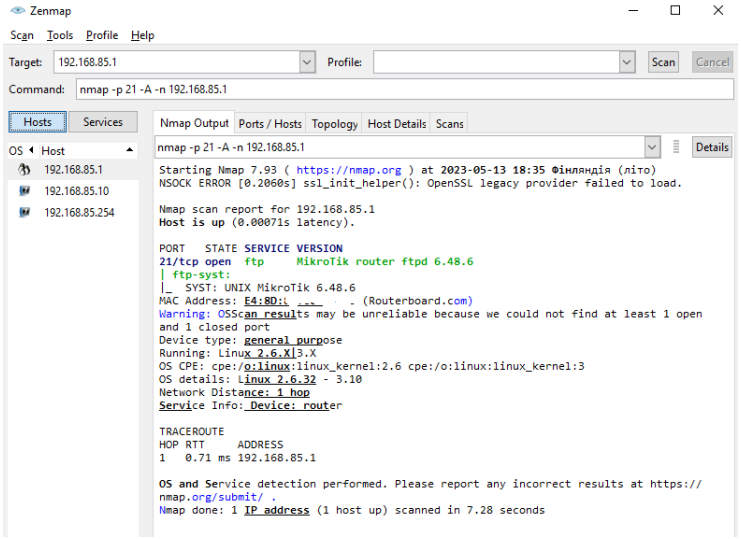


Рисунок 6.15 – Приклад ідентифікації служби додатку

11. Ще раз виконати ідентифікацію ОС вибраного вузла, додавши детальний вивід результатів (опція –vv):

```
nmap -O -n -vv 192.168.x.200
```

12. Переглянути результати, знайти перелік тестів, які виконуються утилітою, звернути увагу на складність підбору значення поля Initial Sequence Number.

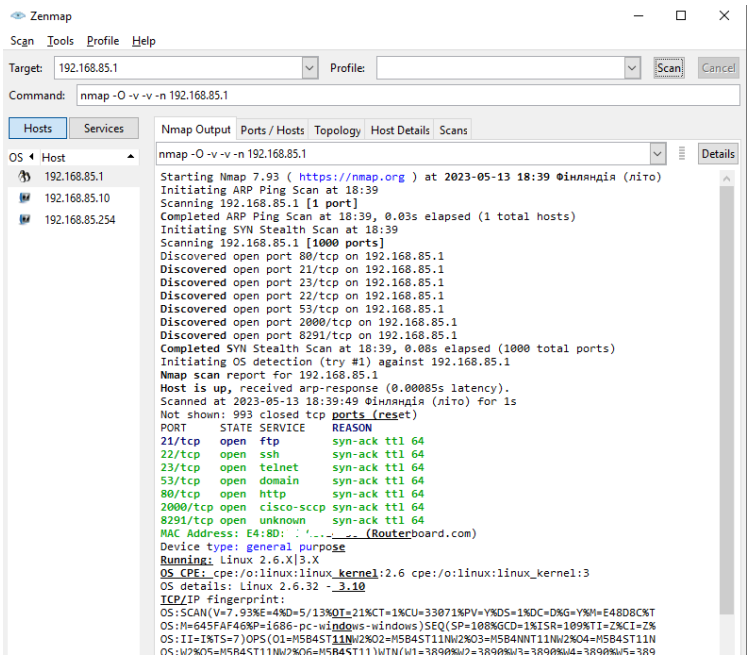


Рисунок 6.16 – Приклад ідентифікації ОС вибраного вузла

13. Виконати ідентифікацію ОС інших вузлів. Порівняти поведінку поля ISN для різних ОС.

14. Запустити процес ідентифікації відкритих портів UDP вибраного вузла (рис.6.16):

```
nmap -sU -n 192.168.x.200
```

15. Переглянути отримані результати.

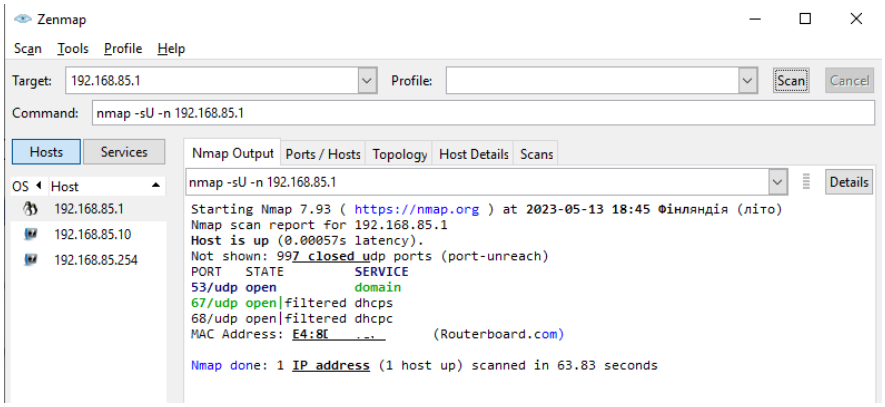


Рисунок 6.17 – Приклад ідентифікації відкритих портів UDP

16. Вибрати кілька вузлів і виконати для них повну інвентаризацію із записом результатів в xml-файл:

```
nmap -A -sS -sU 192.168.x.1-3 -p U:1-500,T:1-500 -n -oX inventory.xml
```

***Примітка***. Опція `-p` вказує діапазон портів, що скануються. Час сканування залежить від заданого діапазона і для даного випадку триває близько 10 хвилин. Для зменшення часу сканування можна скоротити діапазон сканованих портів або відмовитися від сканування портів UDP.

17. Відкрити файл з результатами сканування за допомогою Excel:

18. Видалити всі стовпці до поля IP-адреси (рис.6.18):

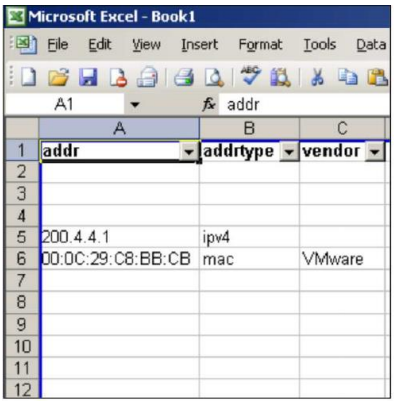


Рисунок 6.18 – Перегляд результатів сканування в Excel

19. Аналогічно, видалити зайві поля, залишивши лише результати інвентаризації (рис.6.19):

| addr | addrtype | protoc | portid | state5 | name | product | version6 | hostname | vendor10 | osfamily | osgen |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 200.4.4.1 | ipv4 | | | | | | | | | | |
| 00:0C:29:C8:BB:CB | mac | | | | | | | | | | |
| | | tcp | 21 | open | ftp | Microsoft ftpd | 5.0 | | | | |
| | | tcp | 25 | open | smtp | Microsoft ESMTP | 5.0.2195.6713 | vm-iss01 | | | |
| | | tcp | 80 | open | http | Microsoft IIS webserver | 5.0 | | | | |
| | | tcp | 119 | open | nntp | Microsoft NNTP Service | 5.0.2195.6702 | | | | |
| | | tcp | 135 | open | msrpc | Microsoft Windows RPC | | | | | |
| | | udp | 135 | open | msrpc | | | | | | |
| | | udp | 137 | open | netbios-ns | Microsoft Windows NT netbios-ssn | | VM-ISS01 | | | |
| | | udp | 138 | open\|filtered | netbios-dgm | | | | | | |
| | | tcp | 139 | open | netbios-ssn | | | | | | |
| | | tcp | 443 | open | https | | | | | | |
| | | tcp | 445 | open | microsoft-ds | Microsoft Windows 2000 microsoft-ds | | | | | |
| | | udp | 445 | open\|filtered | microsoft-ds | | | | | | |
| | | udp | 500 | open\|filtered | isakmp | | | | | | |
| | | | | | | | | | Microsoft | Windows | 95/98/ME |
| | | | | | | | | | Microsoft | Windows | NT/2K/XP |

Рисунок 6.19 – Приклад результатів інвентаризації

20. Видалити зайві рядки, оформити результати, як показано на рис.6.20:

| IP-адрес | addrtype | protoc | portid | state5 | name | product | version5 | hostname | vendor10 | osfamily | osgen |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| 200.4.4.1 | ipv4 | | | | | | | | Microsoft | Windows | 95/98/ME |
| 00:0C:29:C8:BB:CB | mac | | | | | | | | Microsoft | Windows | NT/2K/XP |
| | | | | | | | | | | | |
| | | tcp | 21 | open | ftp | Microsoft ftpd | 5.0 | | | | |
| | | tcp | 25 | open | smtp | Microsoft ESMTP | 5.0.2195.6713 | vm-iss01 | | | |
| | | tcp | 80 | open | http | Microsoft IIS webserver | 5.0 | | | | |
| | | tcp | 119 | open | nntp | Microsoft NNTP Service | 5.0.2195.6702 | | | | |
| | | tcp | 135 | open | msrpc | Microsoft Windows RPC | | | | | |
| | | udp | 135 | open | msrpc | | | | | | |
| | | udp | 137 | open | netbios-ns | Microsoft Windows NT netbios-ssn | | VM-ISS01 | | | |
| | | udp | 138 | open\|filtered | netbios-dgm | | | | | | |
| | | tcp | 139 | open | netbios-ssn | | | | | | |
| | | tcp | 443 | open | https | | | | | | |
| | | tcp | 445 | open | microsoft-ds | Microsoft Windows 2000 microsoft-ds | | | | | |
| | | udp | 445 | open\|filtered | microsoft-ds | | | | | | |
| | | udp | 500 | open\|filtered | isakmp | | | | | | |
| 200.4.4.2 | ipv4 | | | | | | | | Microsoft | Windows | 2003/.NET |
| 00:0C:29:A1:CE:4C | mac | | | | | | | | Microsoft | Windows | NT/2K/XP |
| | | | | | | | | | | | |
| | | tcp | 21 | open | ftp | Microsoft ftpd | | | | | |
| | | tcp | 25 | open | smtp | Microsoft ESMTP | 6.0.2600.2180 | winxp151 | | | |
| | | tcp | 80 | open | http | Microsoft IIS webserver | 5.1 | | | | |
| | | udp | 123 | open | ntp | Microsoft NTP | | | | | |
| | | tcp | 135 | open | msrpc | Microsoft Windows RPC | | | | | |
| | | udp | 137 | open | netbios-ns | Microsoft Windows NT netbios-ssn | | WINXP151 | | | |
| | | udp | 138 | open\|filtered | netbios-dgm | | | | | | |
| | | tcp | 139 | open | netbios-ssn | | | | | | |
| | | tcp | 443 | open | https | | | | | | |
| | | tcp | 445 | open | microsoft-ds | Microsoft Windows XP microsoft-ds | | | | | |
| | | udp | 445 | open\|filtered | microsoft-ds | | | | | | |
| | | udp | 500 | open\|filtered | isakmp | | | | | | |
| 200.4.4.3 | ipv4 | | | | | | | | Linux | Linux | 2.4.X |
| 00:0C:29:B8:E3:54 | mac | | | | | | | | Linux | Linux | 2.5.X |
| | | | | | | | | | Linux | Linux | 2.6.X |
| | | tcp | 22 | open | ssh | OpenSSH | 4.0 | | | | |

Рисунок 6.20 – Приклад оформлення результатів

21. Оформити протокол із скріншотами практичної роботи та завантажити його у відповідну папку на платформі Moodle.

## 6.7. Зміст звіту

1. Результати виконання завдань із скріншотами.

2. Висновки за результатами виконання роботи.

3. Відповіді на контрольні питання.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

## 6.8. Контрольні питання

1. Яку інформацію дозволяє отримати сканування сайту за допомогою команди `nmap -T4 -A -v`?

2. Що виконує вбудована бібліотека *backdoor*?

3. Які операції можна виконати з допомогою бібліотеки *mysql*?

# Лабораторна робота 7

## БАЗОВЕ НАЛАШТУВАННЯ ФАЙРВОЛА В MIKROTIK

**Мета роботи:** Навчитися виконувати базові налаштуванням файрвола маршрутизаторів Mikrotik.

**Зміст.** Вивчаються принципи побудови файрвола, застосування правил за замовчуванням, створення і використання власних правил.

### 7.1. Загальні принципи побудови файрвола

Міжмережевий екран, він же брандмауер або файрвол – найважливіша служба маршрутизатора, яка відповідає як за його безпеку, так і за безпеку всієї мережі. Розберемося з базовим налаштуванням файрвола.

Пакети, що передаються мережею, рухаються вздовж ланцюжка правил у напрямку зверху вниз. Якщо поточний пакет підпадає під якесь правило, він зупиняється на цьому етапі і не рухатиметься далі. Тому в першу чергу варто прописувати правила, які стосуються якомога більшого обсягу трафіку. Це дозволить знизити навантаження на пристрій. Наприклад, одним із перших правил можна дозволити пакети вже встановлених або пов'язаних з'єднань, раніше дозволених іншим правилом. Наново проганяти такі дані через усі правила не потрібно.

Ланцюжки правил в Mikrotik:

1. **Input** – пакети, які були надіслані на маршрутизатор.
2. **Forward** –пакети, які проходять через маршрутизатор.
3. **Output** – пакети, надіслані з маршрутизатора.

Існує два основних типи налаштування файрвола: **нормально відкритий** файрвол та **нормально закритий**. У першому випадку дозволено все, що явно не заборонено, у другому – заборонено все, що явно не дозволено. Обидва ці підходи зазвичай гармонійно поєднуються в рамках одного пристрою. Нормально відкритий файрвол застосовується для внутрішніх мереж і вихідних з'єднань, а нормально закритий для всіх інших з'єднань із зовнішніх мереж. Це

дозволяє досягти простоти та зручності для власних користувачів та водночас надійно захиститися від зовнішнього світу. Будь-яке нове з'єднання спочатку перебуває у статусі **нового (new)** і ми маємо прийняти рішення: схвалити його чи заборонити. Для цього завдання і призначені створювані нами правила, у нормально закритому файрволі ми описуємо, які з'єднання та за якими критеріями слід дозволити. Якщо пакет не відповідає жодному правилу, то ми забороняємо таке з'єднання.

Після того, як з'єднання було встановлено – воно переходить у стан **встановлено (established)**, при цьому спеціальний механізм ядра **Connection Tracker** відстежує всі пакети, що проходять, і зіставляє їх з'єднанням, тому ми можемо спокійно оперувати саме станом з'єднання, не замислюючись над технічними тонкощами роботи.

Чи потрібно щоразу відстежувати пакети вже встановленого з'єднання знову і знову ганяючи їх ланцюжком правил? Звичайно ж ні, якщо ми схвалили з'єднання, то в подальшому можемо і повинні схвалювати всі його пакети автоматично, це знижує навантаження на пристрій і дозволяє використовувати різні прийоми, як Port Knocking або протидії перебору паролів.

Найчастіше одні з'єднання, будучи встановленими, створюють інші, які називаються **пов'язаними (related)**, як приклад можна навести добре відомі FTP або PPTP, які мають керуюче з'єднання для передачі даних. Вони можуть використовувати динамічні порти і тому їх також слід автоматично дозволяти, якщо ми дозволили основне з'єднання.

Тому в будь-якому коректно налаштованому файрволі найпершим правилом має знаходитися дозвільно встановлені і пов'язані з'єднання, а тільки після нього ми вже повинні аналізувати нові підключення.

У маршрутизаторах Mikrotik існує ще один тип з'єднання – **невідстежуване (untracked)**, такі з'єднання не відстежуються **Connection Tracker**, що дозволяє суттєво знизити навантаження на пристрій. Навіщо це може бути потрібно? Припустимо, у фільтрованому нами трафіку є значна частка що належить довіреним вузлам, для яких можна чітко прописати критерії, в

цьому випадку можна позначити їх невідстежуваними і додати їх схвалення в перше правило, до вже встановлених і пов'язаних. Таким чином, одним єдиним правилом ми відразу пропускатимемо переважну частку трафіку, а до витратних операцій аналізу будуть доходити тільки нові пакети. Але тут саме час згадати ще про один стан з'єднання – **недійсне (invalid)**, це пакети, що не є новими, але не належать ніякому встановленому з'єднанню, такі пакети можуть використовуватися для атак і тому їх слід блокувати в першу чергу відразу після того, як ми схвалили встановлені та зв'язані з'єднання і до того, як приступили до аналізу нових пакетів. Ну і завершальним правилом у ланцюжку має стояти **забороняюче**, що відсікає всі з'єднання, які не потрапили під дозвільні правила і не є вже встановленими або пов'язаними з ними.

### 7.2. Конфігурація стандартного файрвола

Насамперед розглянемо конфігурацію, запропоновану виробником за умовчанням. Багато хто вважає її найбільш оптимальною і безпечною, але це не так, конфігурація за умовчанням спрямована на універсальність і сумісність з різними сценаріями користувача, не дозволяючи йому наробити грубих помилок, які можуть фатально позначитися на безпеці. На рис.7.1 показано, де знаходиться меню налаштування файрволу в WinBox.

Всі правила згруповані в два ланцюжки: **input** – для вхідного трафіку самого маршрутизатора та **forward** – для транзитного між інтерфейсами маршрутизатора. Відразу нагадаємо, що маршрутизатори Mikrtoik не мають жорстко заданих "зовнішніх" і "внутрішніх" інтерфейсів, і взагалі всі ці поняття знаходяться виключно у нас в голові, згідно з логічним планом мережі.

У конфігурації за замовчуванням також є два списки інтерфейсів: **WAN** – куди включений порт, налаштований як зовнішній, зазвичай **ether1** і **LAN**, до складу якого входить міст, що об'єднує порти, що залишилися. Ці списки активно застосовуються у правилах.

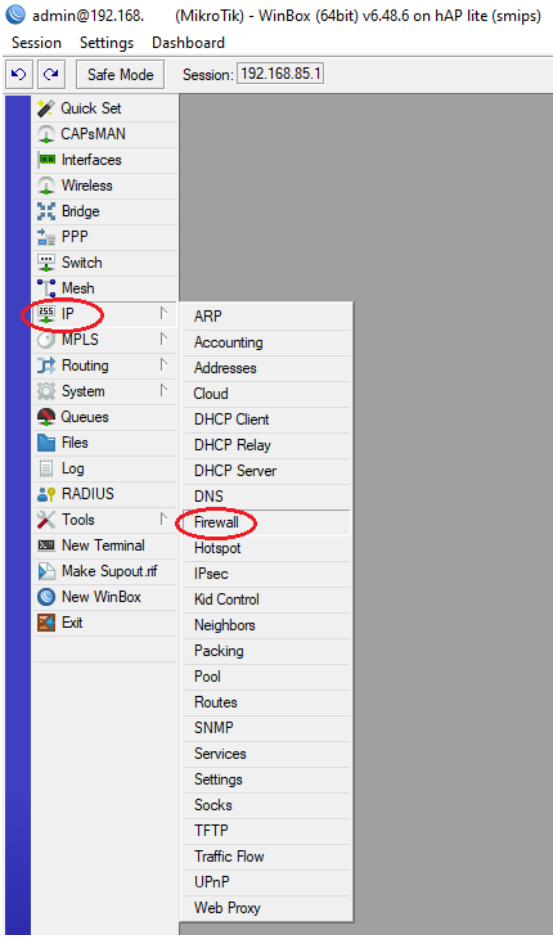


Рисунок 7.1 – Пункт меню Firewall в WinBox

Самих правил небагато, всього 11 штук: 5 для **input** і 6 для **forward** (рис.7.2).

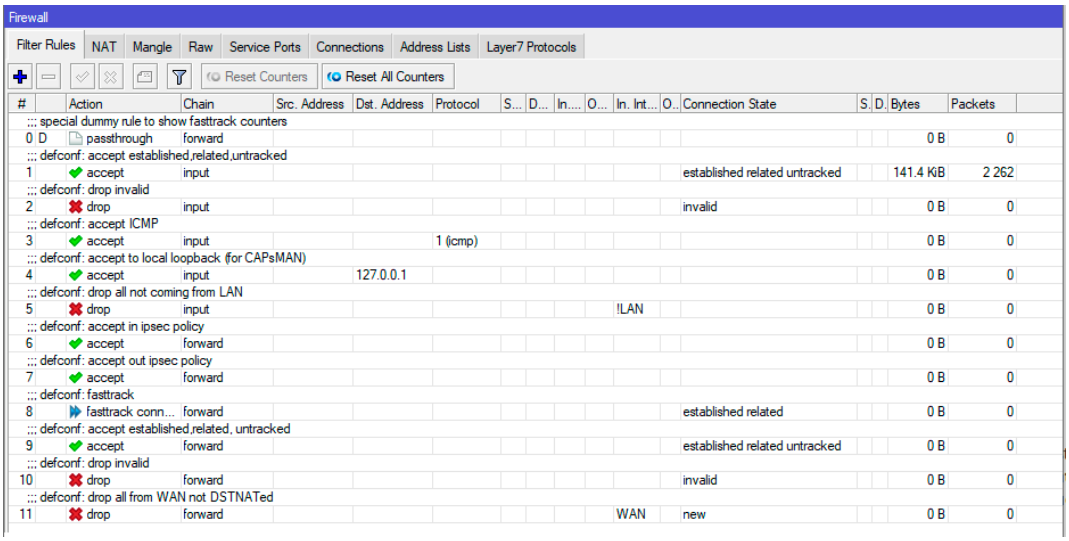


Рисунок 7.2 – Список правил файрвола за замовчуванням

Розглянемо їх докладніше. Правило з нульовим номером до уваги не беремо, це пустушка для виведення лічильників Fasttrack. Почнемо з перших

двох, це класична зв'язка, про яку ми говорили вище: дозволяємо **established**, **related** і **untracked** всім вхідним з'єднанням, без прив'язки до інтерфейсів. Потім забороняємо **invalid**, цілком очевидно, що далі пройдуть лише пакети з'єднань у стані **new**.

```
/ip firewall filter
add chain=input action=accept connection-
state=established,related,untracked comment="defconf: accept
established,related,untracked"
add chain=input action=drop connection-state=invalid
comment="defconf: drop invalid"
```

Третє правило дозволяє вхідний ICMP, багато адміністраторів часто його блокують повністю, щоб пристрій не пінгувався, але так робити не треба, бо протокол ICMP важливий для нормальної роботи мережі. Або фільтрувати необхідні його типи вибірково.

```
add chain=input action=accept protocol=icmp comment="defconf:
accept ICMP"
```

Наступне правило досить специфічне, воно дозволяє вхідні на петльовий інтерфейс, точніше адресу 127.0.0.1, оскільки інтерфейс петлі явно в RouterOS не доступний, а також доданий коментар, що це для CAPsMAN. Його зміст стане зрозумілим трохи нижче.

```
add chain=input action=accept dst-address=127.0.0.1
comment="defconf: accept to local loopback (for CAPsMAN)"
```

І завершує ланцюжок **input** блокуюче правило:

```
add chain=input action=drop in-interface-list=!LAN
comment="defconf: drop all not coming from LAN"
```

Яке блокує вхідні підключення **для всіх**, крім інтерфейсу LAN, а так як локальна петля туди не входить, вище знадобилося окреме правило. Як бачимо, конфігурація трохи надлишкова, але при цьому універсальна. А блокування вхідних за такою широкою маскою – все, крім локальної мережі – дозволяє зберегти належний рівень безпеки, навіть якщо користувач підключить інші зовнішні мережі, скажімо VPN.

А де тут розміщувати свої власні правила? А ось тут:

```
/ip firewall filter
add chain=input action=accept connection-
state=established,related,untracked comment="defconf: accept
established,related,untracked"
add chain=input action=drop connection-state=invalid
comment="defconf: drop invalid"
add chain=input action=accept protocol=icmp comment="defconf:
accept ICMP"
<власні правила>
add chain=input action=accept dst-address=127.0.0.1
comment="defconf: accept to local loopback (for CAPsMAN)"
add chain=input action=drop in-interface-list=!LAN
comment="defconf: drop all not coming from LAN"
```

Тепер про ланцюжок **forward**, найпершими тут тепер додані два правила, що дозволяють транзит з'єднань, які підпадають під дію політик IPsec. Це пов'язано з тим, що IPsec сьогодні застосовується все частіше, але правильно настроїти правила для нього вміють не всі користувачі. Тому з'явилися ці правила, які забезпечують гарантоване проходження будь-яких з'єднань, які використовують IPsec, через маршрутизатор у будь-якому напрямку.

```
add chain=forward action=accept ipsec-policy=in,ipsec
comment="defconf: accept in ipsec policy"
add chain=forward action=accept ipsec-policy=out,ipsec
comment="defconf: accept out ipsec policy"
```

Потім йде правило, що включає **Fasttrack** для всіх встановлених та зв'язаних з'єднань. Якщо коротко, Fasttrack значно знижує навантаження на обладнання, але фактично пускає трафік в обхід файрвола. Але так як фільтрувати **established** і **related** нам не треба, то чому б і не пустити їх коротким шляхом?

```
add chain=forward action=fasttrack-connection connection-
state=established,related comment="defconf: fasttrack"
```

Далі подібно: дозволяємо **established, related** і **untracked**, забороняємо **invalid**. Для всіх напрямків без розбору.

```
add chain=forward action=accept connection-
state=established,related,untracked comment="defconf: accept
established,related, untracked"
add chain=forward action=drop connection-state=invalid
comment="defconf: drop invalid"
```

Нарешті, забороняюче правило:

```
add chain=forward action=drop connection-state=new connection-nat-
state=!dstnat in-interface-list=WAN comment="defconf: drop all
from WAN not DSTNATed"
```

Воно забороняє транзит лише **новим** з'єднанням, лише із **зовнішнього списку** інтерфейсів, крім з'єднань, що пройшли через DNAT. Оскільки DNAT зазвичай використовується для прокидання портів, то таким чином ми уникаємо необхідності створювати для них дозвільні правила.

Так, знову дуже загальні критерії, але в стані абсолютної невизначеності можливих конфігурацій це, мабуть, найкраще рішення. Особливо якщо маршрутизатор буде використовуватися разом з UPnP, де мережні програми можуть самостійно прокидати довільні порти.

Тому ще раз повторимося, що конфігурація за замовчанням не є найоптимальнішою або найбезпечнішою, але вона найбільш універсальна, одночасно закриваючи різні сценарії використання маршрутизатора.

Куди додавати власні правила? Та ось сюди:

```
add chain=forward action=accept connection-
state=established,related,untracked comment="defconf: accept
established,related, untracked"
add chain=forward action=drop connection-state=invalid
comment="defconf: drop invalid"
<власні правила>
```

```
add chain=forward action=drop connection-state=new connection-nat-
state=!dstnat in-interface-list=WAN comment="defconf: drop all
from WAN not DSTNATed"
```

Чи варто залишати та використовувати конфігурацію за замовчанням? Так, якщо ми ще не впевнені у власних силах, погано розуміємо роботу міжмережевого екрану або нам потрібно просто налаштувати маршрутизатор для типових завдань.

### 7.3. Власна конфігурація файрвола

Серед досвідчених користувачів Mikrotik прийнято скидати пристрій і налаштовувати конфігурацію з нуля. Для цього ми використовуємо наступну конфігурацію файрволу, вона багато в чому повторює стандартну, але має ряд відмінностей. У плані іменування інтерфейсів ми також використовуватимемо листи WAN і LAN, з таким самим набором інтерфейсів.

При побудові даного набору правил виходили з міркувань, що адміністратор представляє як влаштована його мережу, яким службам необхідний доступ ззовні, які є зовнішніми, які внутрішніми. При цьому намагалися максимально уникати широких критеріїв у правилах, оскільки "прохідний двір", навіть начебто в досить безпечних речах, таких як IPsec або DNAT може іноді зіграти злий жарт.

Правил у нашому варіанті менше, всього 9, з них 4 для **input** і 5 для **forward** (рис.7.3).

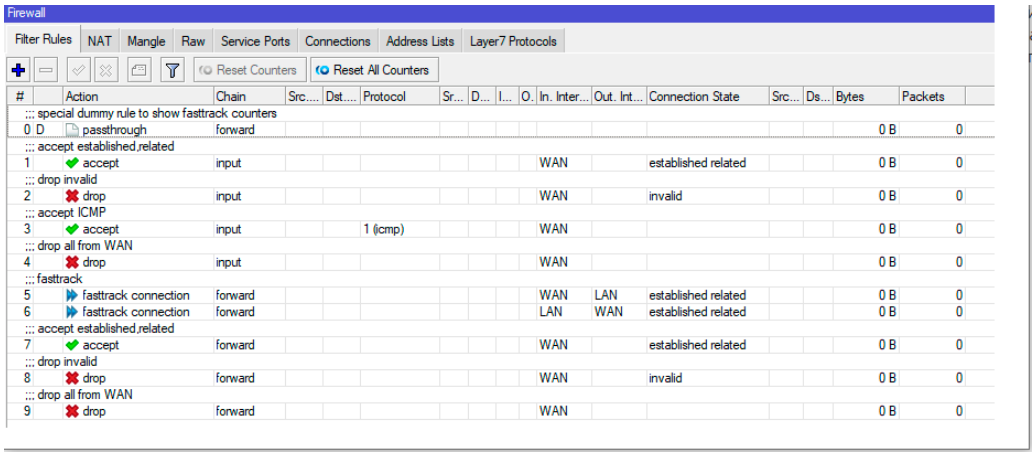


Рисунок 7.3 – Налаштування власних правил

Набір правил для **input** наведемо відразу і повністю, тут той же класичний набір, але за одним винятком: ми знаємо, які мережі є зовнішніми і фільтруємо тільки підключення з них. Це спрощує конфігурацію та знижує навантаження. При цьому в кожному правилі ми чітко вказуємо список вхідних інтерфейсів.

```
/ip firewall filter
add action=accept chain=input connection-state=established,related
in-interface-list=WAN comment="accept established,related"
add action=drop chain=input connection-state=invalid in-interface-
list=WAN comment="drop invalid"
add action=accept chain=input in-interface-list=WAN protocol=icmp
comment="accept ICMP"
<власні правила>
add action=drop chain=input in-interface-list=WAN comment="drop
all from WAN"
```

Ми також дозволяємо **established** і **related**, забороняємо **invalid**, дозволяємо ICMP. Можна помітити, що у нас немає **untracked**, це свідоме рішення, якщо ми будемо використовувати з'єднання, що не відстежуються – то завжди можемо додати цей параметр, але зробимо це усвідомлено. Власні правила ми можемо додавати вище забороняючого всю решту трафіку із зовнішніх інтерфейсів.

Також слід зазначити дещо іншу політику блокування решти трафіку. Стандартна конфігурація підходить досить радикально, забороняючи все, крім LAN. Ми ж використовуємо більш м'який підхід і блокуємо лише зовнішні мережі – WAN. Це дозволяє зробити конфігурацію простішою і не прописувати додаткові правила для того ж CAPsMAN, при цьому ми пам'ятаємо, що адміністратор знає, які мережі є внутрішніми, а які ні, і контролює актуальність списків.

Перейдемо до **forward**, у нашій базовій конфігурації правил для IPsec немає, знову ж таки усвідомлено, через їхню непотрібну надмірність. Як з'явиться IPsec – так і створимо правила, а просто так нема чого їм працювати.

А ось до **Fasttrack** у нас інший підхід. У стандартній конфігурації він включений для всіх **established** і **related**, незалежно від напрямку. Це чудово розвантажує маршрутизатор, але згодом у нас неминуче з'являться тунелі, VPN, інші мережі і Fasttrack там частіше заважатиме, ніж буде допомагати, часом призводячи до несподіваних результатів. Тому чітко вказуємо, що застосовуємо короткий шлях тільки для з'єднань з локальної мережі назовні та ззовні до локальної мережі.

Часто можна навіть відійти від використання списків та просто вказати інтерфейси.

```
add action=fasttrack-connection chain=forward connection-
state=established,related in-interface-list=WAN out-interface-
list=LAN comment="fasttrack"
add action=fasttrack-connection chain=forward connection-
state=established,related in-interface-list=LAN out-interface-
list=WAN
```

До речі, звертаємо увагу, що коментар стоїть лише в одного правила першого. Це зроблено спеціально. Для чого? Подивимося на скріншот вище: там коментар відокремлює одразу два правила.

Ну і далі все те саме. Дозволяємо **established** і **related**, забороняємо **invalid** і обов'язково вказуємо, що саме для зовнішніх мереж, нижче забороняємо все інше, також для зовнішніх інтерфейсів.

```
add action=accept chain=forward connection-
state=established,related in-interface-list=WAN comment="accept
established,related"
add action=drop chain=forward connection-state=invalid in-
interface-list=WAN comment="drop invalid"
<власні правила>
add action=drop chain=forward in-interface-list=WAN comment="drop
all from WAN"
```

Що стосується DNAT, то не варто давати доступ до мережі найширшими дозволами. Якщо з'являється прокид – створюємо окреме правило під прокид. Може це і збільшує кількість роботи, але дає більш зрозумілу та безпечну конфігурацію файрволу, коли ми бачимо кому і що дозволено без вивчення додаткових розділів. Але якщо ми використовуєте маршрутизатор будинку і найширший набір програм прокидає порти за допомогою UPnP, то останнє правило дійсно буде зручніше замінити на:

```
add action=drop chain=forward connection-nat-state=!dstnat in-
interface-list=WAN comment="drop all from WAN"
```

Якщо ми налаштовуємо Mikrotik з нуля, у нас не буде виходу в зовнішню мережу. Щоб отримати його, слід налаштувати NAT.

**Налаштування NAT**

У розділі IP і далі Firewall є потрібна нам вкладка NAT. Додамо таке правило:

```
/ip firewall nat
add action=src-nat chain=srcnat out-interface=ether1-wan to-
addresses=[Ввести IP адресу]
```

Це для випадку, якщо IP є постійним. Якщо він динамічний, тоді використовуємо masquerade:

```
add action=masquerade chain=srcnat out-interface=ether1-wan
```

**Перекидання портів**

Щоб зробити прокидання порту rdp через Mikrotik:

```
add action=dst-nat chain=dstnat dst-port=[Номер порту] in-
interface=ether1-wan protocol=tcp to-addresses=[IP адреса] to-
ports=[Номер порту]
```

Не варто робити доступ до rdp відкритим для всієї зовнішньої мережі. По можливості налаштуємо обмеження доступу через ip. Якщо цього не можна зробити, використовуємо доступ через vpn.

Даний набір правил є базовим кістяком нормально налаштованого файрвола і повинен бути за замовчанням на будь-якому пристрої. Вже потім він обростатиме додатковими правилами, але загальний принцип побудови захисту повинен неухильно дотримуватися. Це дозволить досягти високого рівня безпеки при невеликому навантаженні на пристрій.

Щоб відключити файрвол, потрібно просто усунути всі правила, присутні в списку. Це означатиме, що файрвол пропустить усі пакети.

### 7.4. Завдання

1. Виконайте налаштування файрволу за власними правилами та перевірте їх роботу.

2. Прокоментуйте кожне правило виконаного налаштування.

### 7.5. Зміст звіту

1. Власні правила налаштування файрволу з коментарями.

2. Результати перевірки роботи правил.

3. Висновки за результатами виконання роботи.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 7.6. Контрольні питання

1. Чим відрізняються власні правила налаштування файрволу від правил за замовчуванням?

2. Яка послідовність ланцюжка правил для файрволу.

3. За допомогою якого правила можна організувати перекидання порті? Поясніть його роботу.

# Лабораторна робота 8

## ПОБУДОВА ВИДІЛЕНОГО VPN-СЕРВЕРА НА RASPBERRY PI

**Мета роботи:** Навчитися будувати VPN Raspberry Pi, щоб отримати безпечний віддалений доступ в локальну мережу.

**Зміст.** Розглядається встановлення та налаштування програмного забезпечення для розгортання протоколу WireGuard на RPi для створення власного VPN-сервера.

### 8.1. Особливості застосування

Установимо WireGuard на RPi, щоб створити власний VPN-сервер для безпечного віддаленого доступу до Інтернету та домашньої мережі (рис.8.1).


Рисунок 8.1 – Загальна структура віддаленого доступу через VPN

WireGuard – це досить новий протокол VPN, який набагато безпечніший і швидший, ніж OpenVPN або IPsec. Для безпечного рішення VPN, WireGuard є одним із найкращих виборів: ми можемо налаштувати власну WireGuard VPN на RPi та підключити всі свої пристрої до сервера, не турбуючись про проблему пропускної здатності чи безпеки даних. Розглянемо, як це зробити.

Програма WireGuard VPN доступна для всіх основних платформ, таких як пристрої Windows, Mac, Linux, Android та iOS, які можемо використовувати для безпечного підключення своїх пристроїв до DIY WireGuard VPN.

Щоб створити власний VPN за допомогою WireGuard, нам знадобиться наступне:

• Raspberry Pi 3 або 4 з блоком живлення та корпусом.

• Карта microSD на 8 ГБ або більше.

• Кабель Ethernet для підключення Raspberry Pi до маршрутизатора.

• USB-миша та клавіатура (опціонально)

**Крок 1. Підготовка SD-карти**

1.Завантажимо інструмент Raspberry Pi Imager на інший комп'ютер і підключимо до нього картку microSD.

2. Запустимо Raspberry Pi Imager і натиснемо Choose OS > Raspberry Pi OS (Other) > Raspberry Pi OS (64-bit).

3. Клацнемо Choose Storage і виберемо картку microSD. Переконаємося, що картка порожня або на ній немає важливих даних.

4. Натиснемо Write. Підтвердимо, коли з'явиться підказка. Це може зайняти деякий час.

5. Після завершення SD-карту буде автоматично витягнуто. Від'єднаємо картку та підключимо її знову.

6. Відкриємо вікно File Explorer або Finder, а потім відкриємо завантажувальний розділ.

7. Створимо два файли: ssh і wpa_supplicant.conf (якщо збираємося використовувати Wi-Fi для підключення до мережі).

8. У файл wpa_supplicant.conf вставимо наступне (змінивши AU на власний код країни, а значення ssid і psk на дані нашого власного маршрутизатора), а потім збережемо його.

```
country=AU
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
       ssid="MyWiFiNetwork"
       psk="aVeryStrongPassword"
       key_mgmt=WPA-PSK
}
```

Файл ssh залишається порожнім без розширення.

Настійно рекомендується підключити RPi до мережі або маршрутизатора за допомогою кабелю Ethernet для підвищення швидкості та безпеки.

**Крок 2. Налаштування та інсталяція PiVPN**

PiVPN – це простий скрипт, який можна використовувати для встановлення та налаштування WireGuard на RPi. Кроки для налаштування такі:

1. Вставимо картку microSD у RPi.

2. Під’єднаємо кабель HDMI до RPi та дисплея, наприклад монітора чи телевізора.

3. Також підключимо USB-клавіатуру та мишку.

4. Увімкнемо живлення, щоб завантажити RPi.

5. Налаштуємо початкове налаштування ОС Raspberry Pi, створимо локальний обліковий запис користувача, інсталюємо оновлення та перезапустимо. Запам'ятаємо локальне ім'я користувача та пароль, які ми створили.

Після перезапуску відкриємо вікно терміналу та виконаємо команду, щоб оновити пакети (рис.8.2):

```
sudo apt update && sudo apt full-upgrade
```

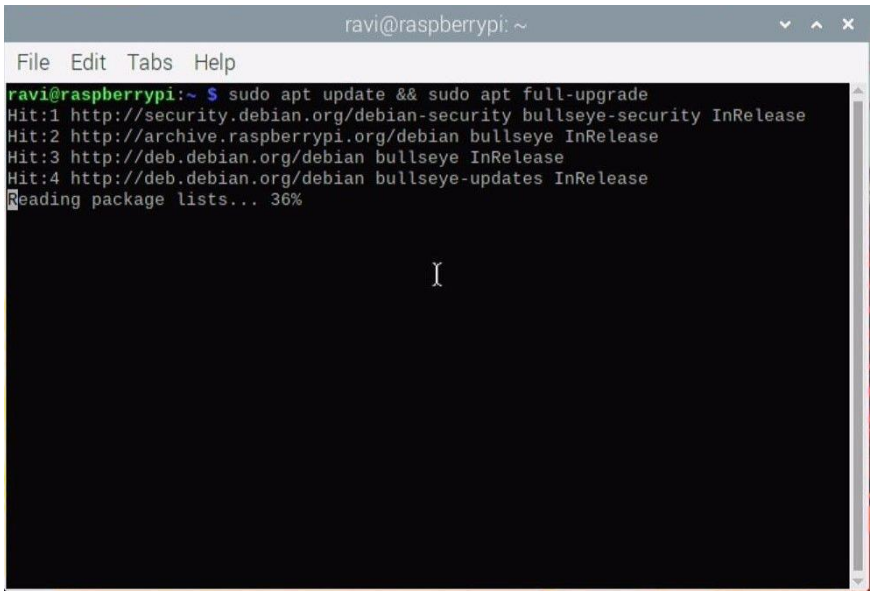


Рисунок 8.2 – Оновлення пакетів

Натиснемо Y для підтвердження (якщо з’явиться підказка), щоб продовжити процес оновлення. Це може зайняти деякий час. Після оновлення пакетів виконаємо наступну команду у вікні терміналу (рис.8.3), щоб розпочати встановлення PiVPN і WireGuard:

```
curl -L https://install.pivpn.io | bash
```

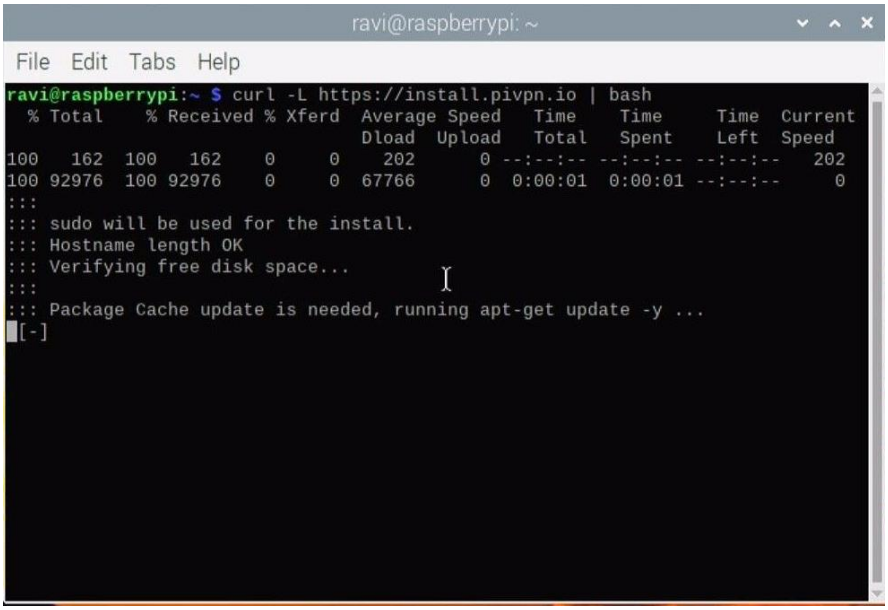


Рисунок 8.3 – Запуск встановлення PiVPN і WireGuard

Буде завантажений і відкриється майстер налаштування PiVPN, який ми використаємо для встановлення WireGuard VPN. Якщо отримаємо повідомлення про помилку curl, переконаємося, що curl встановлено, виконавши наведену нижче команду у вікні терміналу, а потім повторимо спробу:

```
sudo apt install curl -y
```

Після завантаження налаштування PiVPN з’являється вікно майстра налаштування (рис.), у якому відображається повідомлення «This installer wizard will transform your Raspberry Pi into an OpenVPN and WireGuard server!» (рис.8.4).

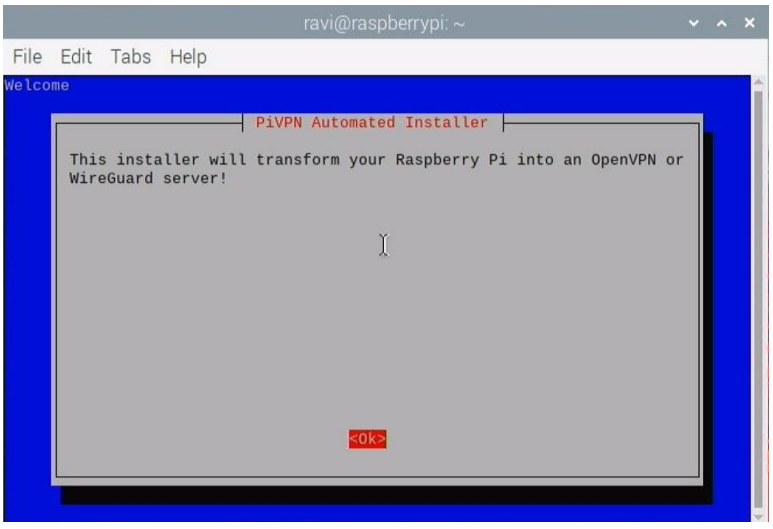


Рисунок 8.4 – Вікно майстра налаштування

Нам потрібно налаштувати або призначити статичну IP-адресу RPi, щоб запобігти будь-якій зміні IP-адреси після втрати живлення або перезавантаження (рис.8.5). Можемо зарезервувати IP-адресу в налаштуваннях DHCP маршрутизатора. Якщо IP у нашому маршрутизаторі зарезервована, вибираємо Yes. Якщо не можемо цього зробити, виберемо No, щоб налаштувати статичну IP-адресу на RPi.

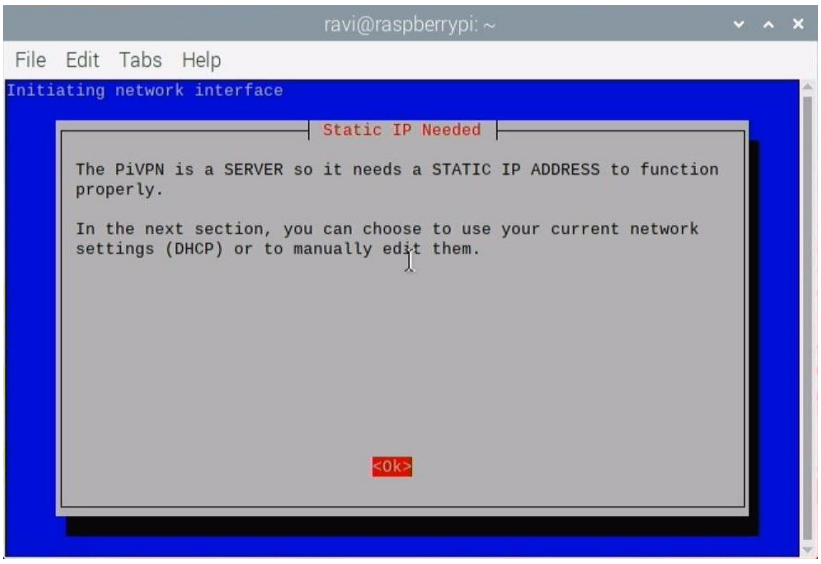


Рисунок 8.5 – Ініціалізація мережевого інтерфейсу

Користувачем за умовчанням є ***pi***, і його має бути видно на цьому екрані. Однак, якщо ми налаштували профіль під час налаштування ОС Raspberry Pi (64-розрядна), наше ім’я користувача з’явиться тут. Використовуємо клавіші із стрілками, щоб виділити його, і пробіл, щоб вибрати його, а потім натискаємо клавішу Enter або вибираємо OK (рис.8.6).

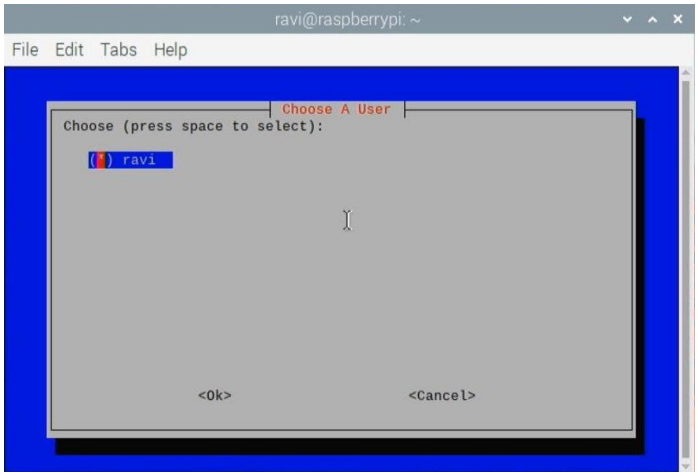


Рисунок 8.6 – Вибір користувача

З доступних параметрів виберемо WireGuard і натиснемо OK або клавішу Enter (рис.8.7).

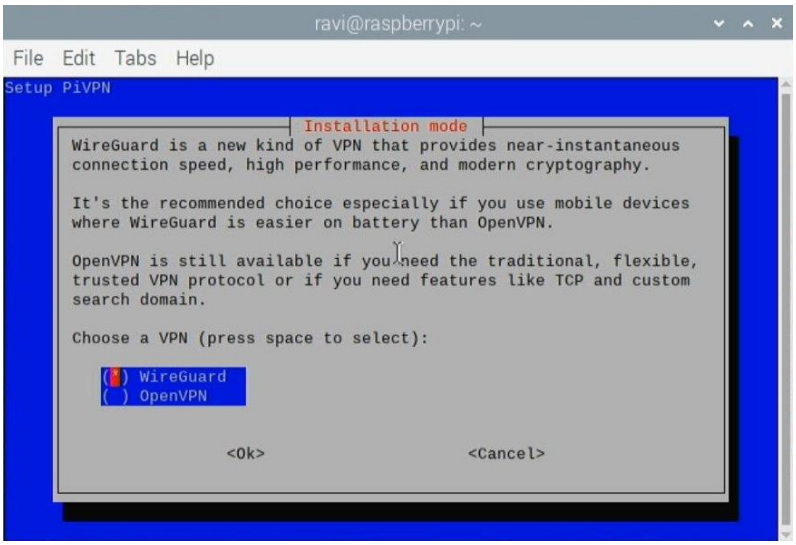


Рисунок 8.7 – Вибір типу VPN

Установимо необхідні оновлення, якщо з'явиться запит. В іншому випадку на екрані встановлення WireGuard виберемо Yes. Почнеться встановлення сервера WireGuard VPN на RPi. Не вносимо жодних змін, коли нас запитають. Збережемо все за замовчуванням, наприклад порт за замовчуванням 51820 (рис.8.8), якщо не хочемо використовувати інший.

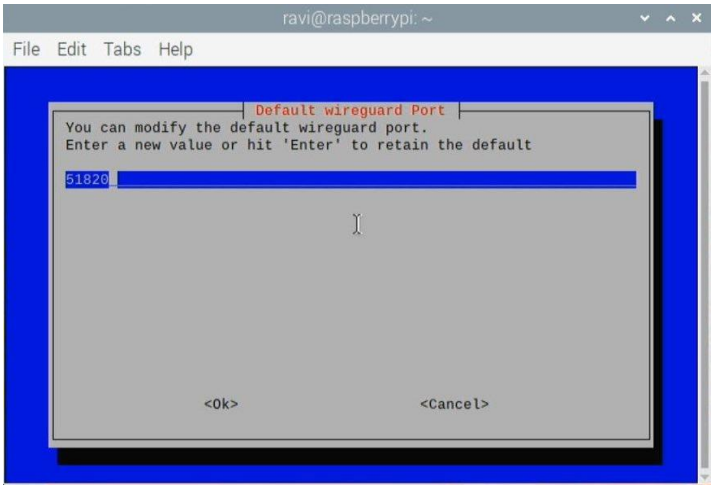


Рисунок 8.8 – Вибір порта

Підтвердимо стандартний порт і виберемо Yes. На наступному екрані нам потрібно вибрати постачальника DNS (рис.8.9). Ми можемо вибрати будь-який залежно від уподобань і місця розташування. Якщо використовуємо DNS-сервер

для блокування реклами, наприклад Pi-hole, нам треба вибрати Custom, а потім ввести DNS-адресу вручну.

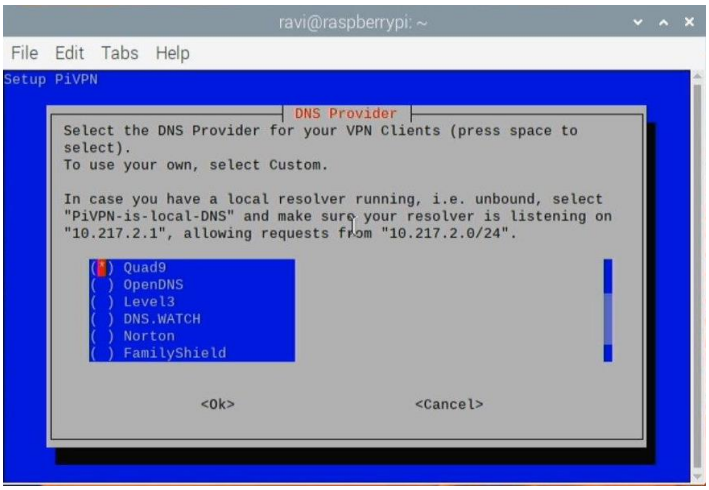


Рисунок 8.9 – Вибір постачальника DNS

Підтвердимо вибір вибраного DNS-сервера та виберемо Yes або натиснемо Enter, щоб продовжити. Нам буде показано два варіанти: можемо використовувати загальнодоступну IP-адресу або загальнодоступний DNS.

Виберемо параметр …use this public IP, якщо наш постачальник послуг призначив нам статичну IP-адресу.

В іншому випадку вибираємо DNS Entry, щоб використовувати загальнодоступний DNS (для динамічної IP-адреси) і налаштовуємо динамічний DNS (рис.8.10). Можемо вибрати з найкращих безкоштовних постачальників динамічних DNS. Наприклад, можемо перейти на no-ip і створити там безкоштовне домене ім’я.

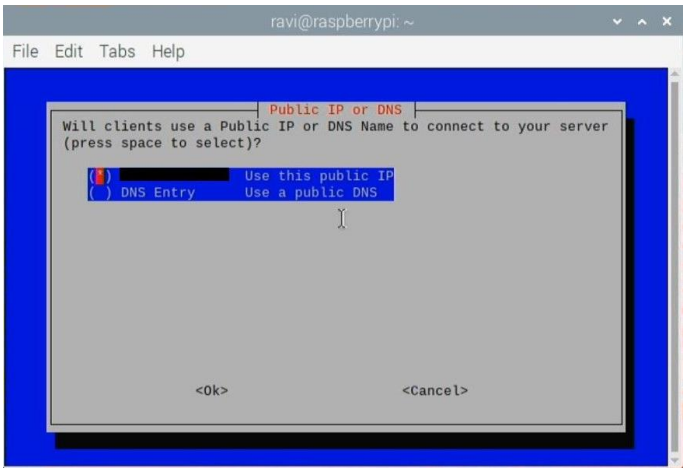


Рисунок 8.10 – Вибір загальнодоступних IP-адреси або DNS

Підтвердимо загальнодоступну IP-адресу або DNS, вибравши Yes. На цьому етапі будуть згенеровані ключі сервера. На наступному кроці потрібно виконати автоматичне оновлення, щоб встановити необхідні пакети (рис.8.11).

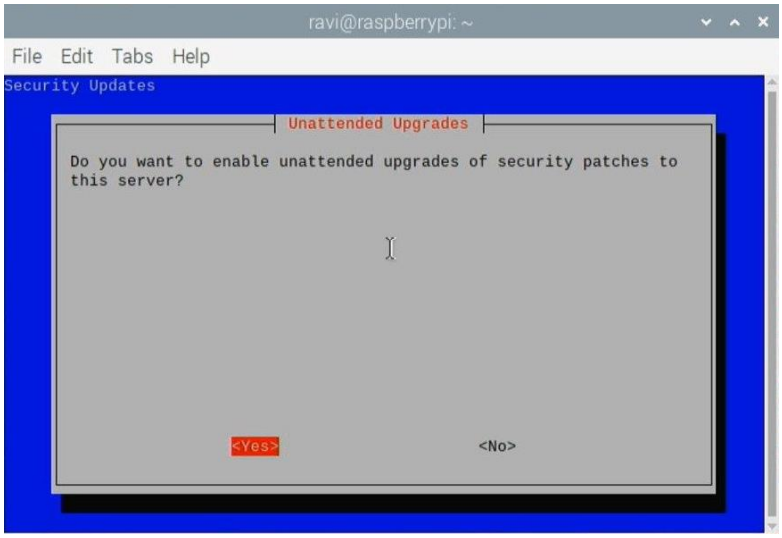


Рисунок 8.11 – Запит на виконання оновлення

На цьому етапі ми завершили встановлення WireGuard VPN. Потрібно перезавантажити RPi, щоб зміни набули чинності (рис.8.12).

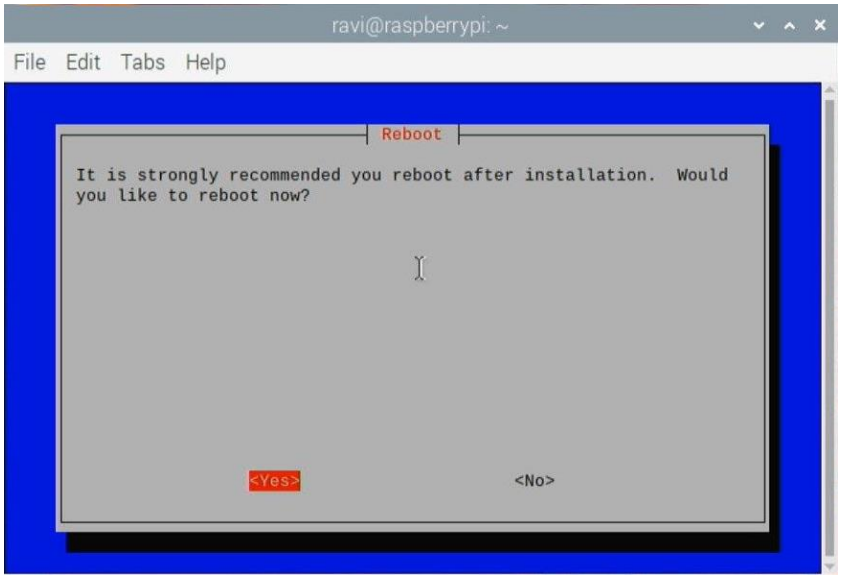


Рисунок 8.12 – Запит на перезавантаження

**Крок 3. Підключаємо пристрої до Raspberry Pi WireGuard VPN**

Щоб підключитися до нашої RPi WireGuard VPN, ми повинні створити профіль. У вікні терміналу введемо таку команду (рис.8.13):

```
sudo pivpn add
```

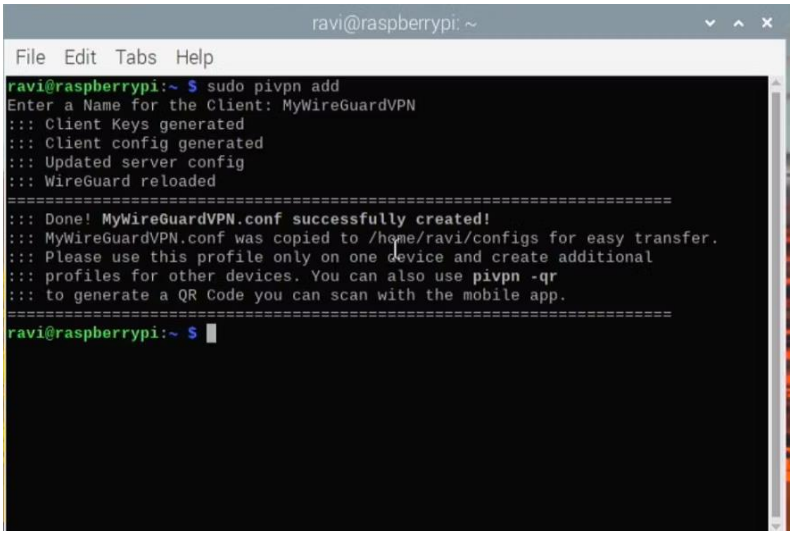


Рисунок 8.13 – Підключення WireGuard VPN

Введемо дані свого профілю. Можете назвати його як завгодно, наприклад «MyWireGuardVPN». Це налаштує наш профіль. Файл конфігурації для підключення можна знайти в `/home/pi/configs` (рис.8.14).


Рисунок 8.14 – Перегляд файла конфігурації

Можемо використовувати цей файл конфігурації для з’єднання WireGuard або створити QR-код для безпечного з’єднання WireGuard VPN. Для цього нам треба встановити додаток WireGuard VPN на свій пристрій Android або iOS. Щоб згенерувати QR-код, запустимо наступну команду у вікні терміналу на RPi.

```
pivpn -qr MyWireGuardVPN
```

У програмі WireGuard на пристрої Android або iOS торкніться значка + (плюс) і виберемо SCAN FROM QR CODE, щоб відсканувати QR-код. Введемо назву профілю та торкнемося Save.

Деякі служби VPN пропонують клієнтську програму для користувачів RPi. Це схоже на використання VPN-клієнта на ПК з Windows і не вимагає завантаження файлів конфігурації OpenVPN.

Відомі три мережі VPN, які пропонують клієнтську програму RPi. Установивши її, можемо вибрати сервер, через який бажаємо отримати доступ до Інтернету, і підключитися до нього.

• ProtonVPN – VPN має сервери в 67 країнах, придатні для потокового передавання, P2P і доступу до Tor.

• NordVPN – одна з найбільших мереж VPN із серверами в 60 країнах.

• Private Internet Access – швидкий VPN із серверами в понад 80 країнах.

Ці рішення ідеально підходять, якщо треба швидкий доступ до VPN.

Швидкість WireGuard залежатиме від швидкості нашої мережі.

Ми розглянули, як налаштувати WireGuard на Raspberry Pi 3 або 4 і підключити наші мобільні пристрої до безпечної VPN. Тепер можемо використовувати власний VPN-сервер Raspberry Pi WireGuard для безпечного віддаленого доступу до Інтернету та домашньої мережі. Не треба платити за підписку або реєстрації даних. Це також забезпечує конфіденційність постачальника послуг Інтернету та допомагає безпечно отримувати доступ до вмісту з будь-якого місця.

### 8.2. Завдання

Налаштувати VPN на RPi, згідно наведеної інструкції.

### 8.3. Зміст звіту

1. Перевірка роботи VPN.

2. Висновки за результатами виконання роботи.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 8.4. Контрольні питання

1. Чому для побудови VPN на RPi вибрано WireGuard?

2. Що таке динамічний DNS?

3. Для чого використовується PiVPN?

## Лабораторна робота 9
## ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ IDS НА SNORT

**Мета роботи:** Отримати відомості про те, як здійснюється захист за допомогою систем виявлення та запобігання вторгнень. Навчитися використовувати Snort.

**Зміст.** Розглядається налаштування та використання систем виявлення та запобігання вторгнень на основі Snort.

### 9.1. Архітектура Snort

Систему виявлення вторгнень Snort за способом моніторингу системи можна віднести як до вузлової, так і до мережевої системи в залежності від параметрів налаштування. Зазвичай вона захищає певний сегмент локальної мережі від зовнішніх атак з Інтернету. Система Snort виконує протоколювання, аналіз, пошук за вмістом, а також широко використовується для активного блокування або пасивного виявлення цілого ряду нападів і зондувань.

Snort здатний виявляти:

- поганий трафік
- використання експлойтів (виявлення Shellcode)
- сканування системи (порти, ОС, користувачі тощо)
- атаки на такі служби як Telnet, FTP, DNS тощо
- атаки DoS/DDoS
- атаки, пов'язані з Web-серверами (cgi, php, frontpage, iss тощо)
- атаки на бази даних SQL, Oracle тощо
- атаки через протоколи SNMP, NetBios, ICMP
- атаки на SMTP, imap, pop2, pop3
- різні Backdoors
- web-фільтри (частіше використовується для блокування порно контенту)
- віруси

Основу Snort становить движок, що складається з п'яти модулів (рис.9.1):

- ***Сніфер пакетів***: даний модуль відповідає за захоплення переданих через мережу даних для подальшої їх передачі на декодер. Робить він це за допомогою бібліотеки DAQ (Data AcQuisition). Працювати даний сніфер може "в розриві" (inline), в пасивному режимі (passive) або читати мережеві дані з заздалегідь підготовленого файла.
- ***Декодер пакетів***: даний модуль займається розбором заголовків захоплених пакетів, розбором пакетів, пошуком аномалій і відхилень від RFC, аналізом TCP-прапорів, винятком окремих протоколів з подальшим аналізом та іншою аналогічною роботою. Фокусується даний декодер на стеку TCP/IP.
- ***Препроцесори***: якщо декодер розбирає трафік на 2-му і 3-му рівні еталонної моделі, то препроцесори призначені для більш детального аналізу і нормалізації протоколів на 3-му, 4-му і 7-му рівнях. Серед найпопулярніших препроцесорів можна назвати frag3 (робота з фрагментованим трафіком), stream5 (реконструкція TCP-потоків), http_inspect (нормалізація HTTP-трафіку), DCE/RPC2, sfPortscan (застосовуються для виявлення сканування портів) і різні декодери для протоколів Telnet, FTP, SMTP, SIP, SSL, SSH, IMAP тощо.
- ***Движок виявлення атак***: даний движок складається з двох частин. Конструктор правил збирає безліч різних вирішальних правил (сигнатур атак) в єдиний набір, оптимізований для подальшого застосування підсистемою інспекції захопленого і обробленого трафіку в пошуках тих чи інших порушень.
- ***Модуль виведення***: за фактом виявлення атаки Snort може видати (записати або відобразити) відповідне повідомлення в різних форматах - файл, syslog, ASCII, PCAP, Unified2 (двійковий формат для прискореного і полегшеного оброблення).

На рис.9.1 показане просте графічне представлення проходження даних через модулі Snort:

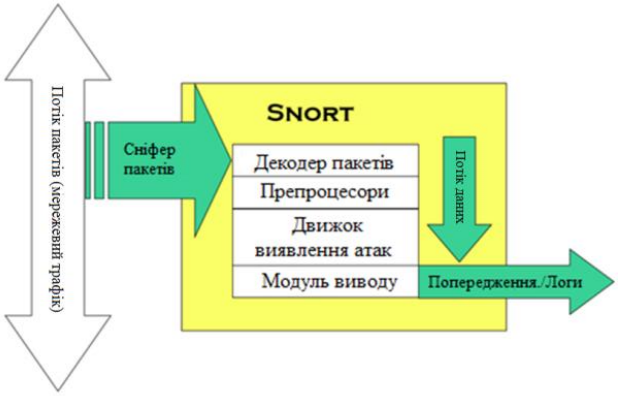


Рисунок 9.1 – Архітектура Snort

## 9.2. Встановлення та налаштування Snort на ОС Raspberry

В першій частині розглянемо приклад встановлення та початкового налаштування Snort для ОС Raspberry. А далі розглянемо конфігурацію мережевого інтерфейса, рекомендованого для Snort.

Для початку перевіримо та встановимо оновлення для ОС:

```
sudo apt update && sudo apt upgrade -y
```

- `apt update` – оновлення індекса пакетів
- `apt upgrade` – оновлення пакетів

Встановлюємо обов'язкові пререквізити для Snort:

```
sudo apt -y install build-essential libpcap-dev libpcre3-dev libdumbnet-dev zlib1g-dev
```

- `build-essential` – надає інструменти, необхідні для збирання пакетів Debian;
- `libpcap-dev` – бібліотека для захоплення мережевого трафіка, необхідна для Snort;
- `libpcre3-dev` – бібліотека функцій для підтримки регулярних виразів, необхідна для Snort;
- `libdumbnet-dev` – бібліотека, ще відома як libdnet-dev, яка надає спрощений, портативний інтерфейс для різних низькорівневих мережевих процедур;
- `zlib1g-dev` – бібліотека, яка реалізує метод стискання deflate, необхідний для Snort.

Встановлюємо необхідні для DAQ парсери:

```
sudo apt -y install bison flex
```

- `bison` – генератор аналізаторів синтаксису (parser) виразів;
- `flex` – інструмент для генерації програм, що розпізнають задані зразки в тексті.

Встановлюємо додаткові (рекомендовані) програми:

```
sudo apt -y install liblzma-dev openssl libssl-dev iptables-dev
libnfnetlink-dev libmnl-dev libnet1-dev libnetfilter-queue-dev
pkg-config autotools-dev checkinstall cmake cpputest
```

Для збирання документації, що не є обов'язковим, необхідно встановити наступні пакети:

```
sudo apt -y install w3m dblatex asciidoc source-highlight
```

Для підтримки HTTP/2 трафіка необхідно встановити біблиотеку Nghttp2: HTTP/2 C-бібліотека, яка реалізує алгоритм стискання заголовків HPAC.

Встановлюємо її з актуального первинного коду (рекомендується перевірити версію при завантаженні):

```
sudo apt install -y autoconf libtool pkg-config
cd ~/snort_src
wget
https://github.com/nghttp2/nghttp2/releases/download/v1.53.0/nghtt
p2-1.53.0.tar.gz
tar -xzvf nghttp2-1.53.0.tar.gz
rm nghttp2-1.53.0.tar.gz
cd nghttp2-1.53.0
autoreconf -i --force
automake
autoconf
./configure --enable-lib-only
make
sudo make install
```

**Збирання DAQ і Snort з первинного коду**

Створимо каталог `snort_src`, куди збережемо архіви, які завантажуємо, з програмами: `mkdir ~/snort_src` Snort використовує бібліотеку DAQ, яку можна завантажити з офіційного сайта Snort. Завантажуємо архів з первинним кодом, створюємо бінарний пакет за допомогою `checkinstall` і встановлюємо його:

```
cd ~/snort_src/
wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz -O
daq-2.0.7.tar.gz
tar -xvzf daq-2.0.7.tar.gz
rm daq-2.0.7.tar.gz
cd daq-2.0.7
./configure
make
sudo checkinstall
sudo dpkg -i daq_2.0.7-1_amd64.deb
```

Оновлюємо шляхи для бібліотек:

```
sudo sh -c "echo >> /etc/ld.so.conf /usr/lib"
sudo sh -c "echo >> /etc/ld.so.conf /usr/local/lib"
sudo ldconfig
```

Далі встановлюємо Snort з первинного коду:

```
cd ~/snort_src/
wget https://www.snort.org/downloads/snort/snort-2.9.17.tar.gz -O
snort-2.9.17.tar.gz
tar -xvzf snort-2.9.17.tar.gz
rm snort-2.9.17.tar.gz
cd snort-2.9.17
./configure --enable-sourcefire
make
sudo checkinstall
sudo dpkg -i snort_2.9.17-1_amd64.deb
```

У випадку будь-яких помилок під час збирання необхідно їх усунути перш ніж рухатися далі.

Створюємо символьне посилання

```
sudo ln -s /usr/local/bin/snort /usr/sbin/snort
sudo ldconfig
```

Варто перевірити, чи правильно встановився Snort. Для цього досить запустити команду `snort -V`. Якщо команда виведе приблизно наступне (номер версії може відрізнятися), то це означатиме, що Snort успішно встановлений:

```
   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.9.0 GRE (Build 56)
   ''''    By Martin Roesch & The Snort Team:
http://www.snort.org/contact#team
           Copyright (C) 2014-2016 Cisco and/or its affiliates.
All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.6.2
           Using PCRE version: 8.35 2014-04-04
           Using ZLIB version: 1.2.8
```

Також за допомогою команди `snort --daq-list` можна подивитися список доступних DAQ-модулів.

**Усунення можливих помилок**

Під час виконання команди `checkinstall` можуть виникнути помилки такого вигляду:

```
ranlib: could not create temporary file whilst writing archive: No
more archived files
```

які можна виправити одним з наступних способів:

встановленням і видаленням програм за допомогою `make`:

```
make install
make uninstall
```

додаванням вручну каталогів, необхідних для збирання і встановлення пакета, які не може створити `checkinstall` самостійно:

```
sudo mkdir /usr/local/lib/snort_dynamicengine
sudo mkdir /usr/local/include/snort
sudo mkdir /usr/local/lib/snort
```

```
sudo mkdir /usr/local/include/snort/dynamic_preproc
sudo mkdir /usr/local/lib/snort/dynamic_preproc
sudo mkdir /usr/local/lib/snort_dynamicpreprocessor
sudo mkdir /usr/local/lib/snort/dynamic_output
sudo mkdir /usr/local/share/doc
```

Після чого знову збираємо і встановлюємо Snort:

```
sudo checkinstall
sudo dpkg -i snort_2.9.17.1_amd64.deb
```

**Конфігурація мережевого інтерфейсу, рекомендована для Snort**

Згідно рекомендаціям manual Snort варто переконатися, що мережева карта не обрізає надто великі пакети (в мережах Ethernet довжина яких перевищує 1518 байт). Для цього виконаємо оптимізацію мережевих інтерфейсів за допомогою утиліти `ethtool`. Якщо треба, встановлюємо `ethtool` (`sudo apt -y install ethtool`) і для налаштування параметрів виконуємо в терміналі наступні команди:

`ethtool -K eth0 rx off` – звільнення ОС від розрахунку контрольних сум TCP/IP (rx-checksumming) для вхідних пакетів;

`ethtool -K eth0 tx off` – звільнення ОС від розрахунку контрольних сум TCP/IP (tx-checksumming) для вихідних пакетів;

`ethtool -K eth0 gso off` – вимкнення функції (generic-segmentation-offload) фрагментації пакетів без участі CPU;

`ethtool -K eth0 gro off` – вимкнення функції (generic-receive-offload) збірки пакетів мережевим інтерфейсом без участі CPU;

`ethtool -K eth0 lro off` – вимкнення функції (large-receive-offload) збірки пакетів мережевим інтерфейсом без участі CPU.

Але, в даному випадку, налаштування будуть залишатися в силі лише до перезавантаження ОС, тому кращим варіантом буде додати команди налаштування мережевого інтерфейса одним з наступних способів:

Відкриваємо файл `/etc/network/interfaces` (`sudo gedit /etc/network/interfaces`) з налаштуваннями конфігурації мережі Ethernet і додаємо в кінець файла наступні команди для мережевого інтерфейса eth0, який Snort буде слухати:

```
# The primary network interface
auto eth0
iface eth0 inet dhcp
post-up ethtool --offload eth0 rx off tx off
post-up ethtool -K eth0 gso off
post-up ethtool -K eth0 gro off
post-up ethtool -K eth0 lro off
```

Або відкриваємо файл `/etc/rc.local` (`sudo gedit /etc/rc.local`) з автозавантаженнями і додаємо наступні рядки до "`exit 0`":

```
ethtool --offload eth0 rx off tx off
ethtool -K eth0 gso off
ethtool -K eth0 gro off
ethtool -K eth0 lro off
```

Поточні значення параметрів можна переглянути за допомогою команди `ethtool -k eth0`. Якщо параметр позначений як [`fixed`], то це означає, що значення параметра не можна змінити за допомогою `ethtool`. Часто багато параметрів позначені [`fixed`], коли ОС запущена у віртуальній машині (наприклад, VMWare або VirtualBox) і гостьова ОС не в змозі змінити параметри хостової ОС.

### 9.3. Налаштування Snort як мережевої системи виявлення вторгнень

Тепер розглянемо конфігурацію Snort в якості мережевої системи виявлення вторгнень (NIDS). Створимо файли і папки, які необхідні для работи Snort, і переглянемо головний конфігураційний файл `snort.conf`.

**Базова конфігурація**

З міркувань безпеки, щоб не запускати Snort із-під суперкористувача, створимо непривілегійованого системного користувача і групу:

```
# Створюємо групу і користувача для Snort:
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
```

Далі створюємо ряд файлів і каталогів, необхідних для роботи Snort, і встановлюємо права доступу для цих файлів:

```
# Створюємо необхідні для Snort каталоги:
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo mkdir /etc/snort/so_rules

# Створюємо файли, в яких будуть зберігатися правила і списки IP:
sudo touch /etc/snort/rules/iplists/black_list.rules
sudo touch /etc/snort/rules/iplists/white_list.rules
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/sid-msg.map

# Створюємо каталоги, де будуть зберігатися лог-файли:
sudo mkdir /var/log/snort
sudo mkdir /var/log/snort/archived_logs

# Змінюємо права доступу:
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /var/log/snort/archived_logs
sudo chmod -R 5775 /etc/snort/so_rules
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules

# Змінюємо власника для каталогів:
sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

Після чого копіюємо наступні конфігураційні файли з розгорнутого архіва Snort в каталог `/etc/snort`:

`classification.config` – описує типи категорії атак, які попередньо встановлені в Snort.

`file_magic.conf` – описуємо файлові сигнатури («магічні числа») для визначення типу файла.

`reference.config` – вміщує посилання на системи ідентифікації атак.

`snort.conf` – конфігураційний файл Snort, в якому зберігаються змінні з налаштуваннями і шляхами до різних ресурсів.

`threshold.conf` – дозволяє налаштувати кількість подій, необхідних для генерації сповіщень (алертів), що може бути корисним у випадку «шумних» правил.

`attribute_table.dtd` – дозволяє Snort використовувати зовнішню інформацію для визначення протоколів і політик.

`gen-msg.map` – вказує Snort який препроцесор використовується яким правилом.

`unicode.map` – надає відповідність між кодуваннями.

Для цього виконаємо наступні команди (перевіряємо версію!):

```
cd ~/snort_src/snort-2.9.17.0/etc/
sudo cp *.conf* /etc/snort
sudo cp *.map /etc/snort
sudo cp *.dtd /etc/snort

cd ~/snort_src/snort-2.9.17.0/src/dynamic-
preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
sudo cp * /usr/local/lib/snort_dynamicpreprocessor/
```

Структура каталогів і файлів Snort повинна виглядати наступним чином, а для її виведення на екран треба виконати команду `tree /etc/snort` (якщо утиліта `tree` не встановлена, то її можна встановити наступним чином:

```
sudo apt install tree:
     /etc/snort
```

```
├── attribute_table.dtd
├── classification.config
├── file_magic.conf
├── gen-msg.map
├── preproc_rules
├── reference.config
├── rules
│   ├── iplists
│   │   ├── black_list.rules
│   │   └── white_list.rules
│   └── local.rules
├── sid-msg.map
├── snort.conf
├── so_rules
├── threshold.conf
└── unicode.map

4 directories, 12 files
```

**Зміна конфігураційних файлів Snort**

Конфігураційний файл Snort `/etc/snort/snort.conf` включає всі налаштування, які використовує Snort в режимі NIDS. Файл для зручності розділений на наступні частини:

1. Встановлення мережевих значень і змінних (Set the network variables).
2. Конфігурація декодерів (Configure the decoder).
3. Конфігурація базового (основного) механізму виявлення (вторгнень) (Configure the base detection engine).
4. Конфігурація динамічно завантажуваних бібліотек (Configure dynamic loaded libraries).
5. Конфігурація препроцесорів (Configure preprocessors).
6. Конфігурація плагінів (Configure output plugins).
7. Налаштування набору правил (Customize your rule set).

8. Налаштування препроцесорів і декодерів набору правил (Customize preprocessor and decoder rule set).

9. Налаштування правил для розшарених об’єктів (Customize shared object rule set).

Три типи змінних, які можна визначити в Snort:

`var` – використовується для присвоювання змінній шляху до файла або директорії і для призначення змінній ip-адрес.

`ipvar` – застосовується до змінних також для вказування ip-адрес, але лише з підтримкою IPv6.

`portvar` – використовується, щоб задати змінні з номерами портів.

Синтаксис визначення змінних:

```
<var | portvar | ipvar> <назва_змінної> <значення_змінної>
```

За допомогою ключового слова `include` підключаються додаткові файли.

Відкриваємо файл (наприклад, `sudo gedit /etc/snort/snort.conf`) і налаштовуємо наступні параметри:

- **Налаштовуємо `Set the network variables`**
  - **Задаємо внутрішню мережу**

Опція `ipvar HOME_NET any` задає діапазон внутрішніх IP-адрес (хост або список хостів) для домашньої мережі, які ми збираємося захищати, і трафік яких Snort буде аналізувати. Значення змінної `$HOME_NET` відповідає стандарту RFC 1918 (адресний простір). Є можливість вказати мережевий інтерфейс: `$<ІМ’Я_МЕРЕЖЕВОГО_ІНТЕРФЕЙСА>_ADDRESS` (наприклад, `ipvar HOME_NET $eth0_ADDRESS`). Значення змінної буде рівне IP-адресі. Найчастіше цей параметр використовується на комп’ютерах, які отримують IP-адресу динамічно. Можна задати маску і вказати кілька IP-адрес через кому (пробіли не допускаються): `ipvar HOME_NET [10.1.1.0/24,192.168.1.0/24]`. За замовчуванням встановлено значення `any`, що може привести до великої кількості помилок першого роду (false positives alerts). В нашому випадку встановимо IP-адресу машини, на якій встановлений Snort:

```
44 # Setup the network addresses you are protecting
```

```
45 ipvar HOME_NET 192.168.0.104   # Необхідно вказати свою IP-
адресу
```

- **Задаємо зовнішню мережу**

Опція `ipvar EXTERNAL_NET any` задає діапазон зовнішніх IP-адрес мереж, з яких виходить загроза. Можна використати логічне заперечення за допомогою символу “!”, наприклад, `ipvar EXTERNAL_NET !$HOME_NET` видаляє із змінної `$EXTERNAL_NET` мережу `$HOME_NET`. Дане значення залишимо за замовчуванням `any`, щоб всі адреси, відмінні від `$HOME_NET`, сприймалися зовнішніми.

```
47 # Set up the external network addresses. Leave as "any" in most
situations
48 ipvar EXTERNAL_NET any
```

Налаштовуємо шляхи до каталогів, які створили раніше. Для цього в наступних рядках

```
104 var RULE_PATH ../rules
105 var SO_RULE_PATH ../so_rules
106 var PREPROC_RULE_PATH ../preproc_rules
113 var WHITE_LIST_PATH ../rules
114 var BLACK_LIST_PATH ../rules
```

змінюємо шляхи, як показано нижче:

```
104 var RULE_PATH /etc/snort/rules
105 var SO_RULE_PATH /etc/snort/so_rules
106 var PREPROC_RULE_PATH /etc/snort/preproc_rules
113 var WHITE_LIST_PATH /etc/snort/rules/iplists
114 var BLACK_LIST_PATH /etc/snort/rules/iplists
```

В даній частині є налаштування змінних IP-адрес для серверів DNS, SMTP, HTTP, SQL, TELNET, SSH, FTP, SIP і портів, які використовуються серверами для конкретних додатків (HTTP, SHELLCODE, ORACLE, SSH, FTP, SIP тощо). Вказавши дані параметри для конкретної мережі, можна більш тонко налаштувати Snort, але у нашому випадку залишимо дані параметри за замовчуванням.

- **Налаштовуємо `Customize your rule set`**

В даному файлі знаходиться список включених файлів виду `include $RULE_PATH/НАЗВА_НАБОРУ_ПРАВИЛ.rules`, які Snort використовує за замовчуванням для імпорту правил. Всі ці рядки, за винятком одного – `include $RULE_PATH/local.rules`, що вказує шлях до файла, де будуть зберігатися наші правила, необхідно закоментувати, так як ми будемо використовувати менеджер правил для Snort – PulledPork, який зберігає всі правила в одному файлі. Закоментувати рядки можна за допомогою команд:

```
# Закоментуємо всі рядки виду include $RULE_PATH
sudo sed -i 's|include \$RULE_PATH|#include \$RULE_PATH|'
/etc/snort/snort.conf
# Розкоментуємо рядок include $RULE_PATH/local.rules
sudo sed -i 's|#include \$RULE_PATH/local\.rules|include
\$RULE_PATH/local\.rules|' /etc/snort/snort.conf
```

Якщо не використовувати PulledPork для управління набором правил, то необхідно вручну завантажити набір правил із сайта Snort і розархівувати їх в каталог `/etc/snort/rules`, звідки Snort їх буде завантажувати при запуску. В такому випадку рядки в конфігураційному файлі `/etc/snort/snort.conf` коментувати не треба.

Після внесення змін в налаштування конфігураційного файла, необхідно виконати наступну команду для перевірки коректності конфігураційного файла:

```
sudo snort -T -c /etc/snort/snort.conf -i eth0
```

де опція `-T` вказує, що треба протестувати поточну конфігурацію Snort, `-c` задає шлях до конфігураційного файла, а `-i` задає мережевий інтерфейс, який Snort буде слухати. У випадку успішного самотестування Snort в кінці виведення будуть наступні рядки:

```
...
Snort successfully validated the configuration!
Snort exiting
```

### 9.4. Створення першого тестового правила для Snort

В поточному конфігураційному файлі `/etc/snort/snort.conf` прописаний єдиний файл з правилами `include $RULE_PATH/local.rules`, який завантажує Snort. В даному файлі вказуються специфічні для нашого середовища правила, а також правила для тестування, але поки що файл порожній. Додамо в даний файл перше тестове правило. Для цього відкриваємо файл `sudo gedit /etc/snort/rules/local.rules` и додаємо в нього правило:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test detected";
GID:1; sid:10000001; rev:001; classtype:icmp-event;)
```

Дане правило означає: для будь-яких ICMP-пакетів з будь-якої мережі в нашу домашню мережу `HOME_NET` генерується попередження `ICMP test detected`. Після того, як внесли зміни у файл, який завантажує Snort, було б добре протестувати конфігураційний файл Snort:

```
sudo snort -T -c /etc/snort/snort.conf -i eth0
```

У випадку успішного тестування побачимо у виводі даної команди наступні рядки про наше додане правило:

```
++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
1 Snort rules read
    1 detection rules
    0 decoder rules
    0 preprocessor rules
1 Option Chains linked into 1 Chain Headers
0 Dynamic rules
++++++++++++++++++++++++++++++++++++++++++++++++++
+-------------------[Rule Port Counts]-------------
     |             tcp     udp    icmp      ip
     |     src       0       0       0       0
     |     dst       0       0       0       0
     |     any       0       0       1       0
     |      nc       0       0       1       0
     |     s+d       0       0       0       0
     +-----------------------------------------------
```

Далі протестуємо дане правило, запустивши Snort і переконавшись, що він генерує попередження при обробленні ICMP-пакетів. Запускаємо Snort із наступними опціями (ключами):

```
sudo snort -A console -q -u snort -g snort -c
/etc/snort/snort.conf -i eth0
```

`-A console` – консольна опція для друку попередження у швидкому режимі на стандартний потік виведення stdout;

`-q` – тихий режим без показування банера і звіту;

`-u snort` – вказуємо кристувача snort із-під якого запускається Snort;

`-g snort` – вказуємо групу snort із-під якої запускається Snort;

`-c /etc/snort/snort.conf` – задаємо шлях до конфігураційного файла snort.conf;

`-i eth0` – вказуємо мережевий інтерфейс, який Snort буде слухати.

Після того, як запустили Snort за допомогою вказаної вище команди, необхідно з іншої машини надіслати ICMP-пакети на мережевий інтерфейс машини, який слухає Snort (наприклад, виконавши команду `ping 192.168.0.104`). В результаті чого в запущеному терміналі із Snort повинні з'явитися приблизно наступні рядки з попередженнями, що означає – Snort успішно працює в режимі мережевої IDS і генерує попередження:

```
02/10-00:48:14.587138  [**] [1:10000001:1] ICMP test detected [**]
[Classification: Generic ICMP event] [Priority: 3] {ICMP}
192.168.0.100 -> 192.168.0.104
02/10-00:48:15.590848  [**] [1:10000001:1] ICMP test detected [**]
[Classification: Generic ICMP event] [Priority: 3] {ICMP}
192.168.0.100 -> 192.168.0.104
02/10-00:48:16.596818  [**] [1:10000001:1] ICMP test detected [**]
[Classification: Generic ICMP event] [Priority: 3] {ICMP}
192.168.0.100 -> 192.168.0.104
02/10-00:48:17.602570  [**] [1:10000001:1] ICMP test detected [**]
[Classification: Generic ICMP event] [Priority: 3] {ICMP}
192.168.0.100 -> 192.168.0.104
```

Щоб зупинити роботу Snort можна натиснути **CTRL+C** в терміналі із запущеним Snort. Всі лог-файли зберігаються в каталозі `/var/log/snort`.

Беручи до уваги, що RPi споживає зовсім мало енергії, його використання разом із Snort буде ефективним рішенням для організації захисту мережі.

### 9.5. Завдання

1. Встановити та налаштувати на RPi Snort.
2. Запропонувати та виконати операції для перевірки роботи Snort.

### 9.6. Зміст звіту

1. Скріншоти результатів встановлення та налаштування Snort.
2. Висновки за результатами виконання роботи.

Звіт в електронному вигляді (бажано у форматі pdf) завантажити у відповідну папку в Moodle.

### 9.7. Контрольні питання

1. Чим відрізняються пасивні і активні IDS?
2. Які задачі виконує Snort?
3. Як працюють правила Snort?
4. Як писати правила для Snort?
5. Як в Snort створювати логи?

## РЕКОМЕНДОВАНА ЛІТЕРАТУРА

**Базова література:**


1. Могильний С.Б. Мікрокомп’ютер Raspberry Pi – інструмент дослідника. – К.: Талком, 2014. – 340 с. (Електронна версія http://isearch.kiev.ua/uk/book/1850-microcomputer-raspberry-pi-tool-researcher)
2. Могильний С.Б. Вбудовані системи програмно-апаратних комплексів обробки інформації: Лабораторний практикум [Електронний ресурс]: навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 172 «Електронні комунікації та радіотехніка» / С.Б.Могильний; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 3,74 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 121 с. – Назва з екрана.
3. Могильний С.Б. Інформаційна безпека при роботі в Інтернеті : навчально-методичний посібник, 2018. / [За редакцією О.В.Лісового та ін.]. -К., 2018, - 105 с. (Електронна версія http://isearch.kiev.ua/uk/book/1954-445000-information-security-when-browsing-the-interne)
4. Кавун С. В. Інформаційна безпека. Навчальний посібник / С. В. Кавун, В. В. Носов, О. В. Манжай. — Харків: Вид. ХНЕУ, 200’. – 352 с.
5. Інформаційна безпека : навчальний посібник / Ю. Я. Бобало, І. В. Горбатий, М. Д. Кіселичник, А. П. Бондарєв, С. С. Войтусік, А. Я. Горпенюк, О. А. Нємкова, І. М. Журавель, Б. М. Березюк, Є. І. Яковенко, В. І. Отенко, І. Я. Тишик ; за заг. ред. д-ра техн. наук, проф. Ю. Я. Бобала та д-ра техн. наук, доц. І. В. Горбатого. – Львів : Видавництво Львівської політехніки, 2019. – 580 с.


**Додаткова література:**


1. Akashdeep Bhardwaj, Varun Sapra. Security Incidents & Response Against Cyber Attacks. Springer, 2021, - 250 p.
2. Інформаційна безпека. Підручник / В. В. Остроухов, М. М. Присяжнюк, О. І. Фармагей, М. М. Чеховська та ін.; під ред. В. В. Остроухова – К.: Видавництво Ліра-К, 2021. – 412 с.

3. Грайворонський, М. В. Безпека інформаційно-комунікаційних систем [Електронний ресурс] : підручник / М. В. Грайворонський, О. М. Новіков. – Київ : Видавнича група BHV, 2009. – 698 с.

**Інформаційні ресурси Інтернету:**

1. Сайт Академії Mikrotik:
   - https://mikrotik.kpi.ua/index.php/courses-list/category-raspberry
   - https://mikrotik.kpi.ua/index.php/courses-list/category-python
2. Персональний сайт викладача: – http://isearch.kiev.ua/
3. Сайт дистанційного навчання на платформі Moodle Академії Mikrotik: – http://iot.kpi.ua/lms/
4. Платформа дистанційного навчання «Сікорський»: – https://www.sikorsky-distance.org/