Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____*Інфокомунікацій*_____
(повна назва)
Кафедра _____*Інформаційно-мережної інженерії*_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____*Реалізація механізмів захоплення та аналізу пакетів у середовищі Linux*_____
(тема)

Виконав:
студент _2_ курсу, групи ____*ІМІм-21-1*____
_____*Чернобровкін О.С.*_____
(прізвище, ініціали)

Спеціальність _*172. Телекомунікації та радіотехніка*_
(код і повна назва спеціальності)
Тип програми _____*освітньо-професійна*_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _*Інформаційно-мережна інженерія*_
(повна назва освітньої програми)

Керівник _____*ст.викл. Твердохліб В.В.*_____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _____*Безрук В.М.*_____
(підпис) (прізвище, ініціали)

2024 р.

Не містить відомостей, заборонених
до відкритого публікування

Керівник _____ /*В.В.Твердохліб*

Студент _____ / *О. С. Чернобровкін*

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*

Кафедра *Інформаційно-мережної інженерії*

Рівень вищої освіти другий (магістерський)

Спеціальність *172. Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *Освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_23_» *жовтня* 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Чернобровкіну Олександру Сергійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Реалізація механізмів захоплення та аналізу пакетів у середовищі Linux*

затверджена наказом університету від _23_ жовтня 2023 р. № 1233 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 січня 2024 р.

3. Вихідні дані до роботи *Дослідити функціонал та механізми, на базі яких реалізовано програмні засоби захоплення та аналізу пакетів. Розглянути можливості ПЗ, яке побудоване на базі Pcap/LibPcap, зокрема, WireShark та аналогічних засобів, а також їх роль у зменшенні інформаційних ризиків. Дослідити можливості та склад пакету LibPcap.Побудувати консольний додаток, що виконує захоплення та аналіз пакетів у середовищі Linux.*

4. Перелік питань, що потрібно опрацювати в роботі *Вступ*
1 . Аналіз існуючих інформаційних ризиків
2. Огляд поширених засобів моніторингу мережевих пакетів
3. Підготовчі операції, необхідні для реалізації процесу сніфінгу
4. Сніфінг пакетів засобами libpcap та їх наступний аналіз
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
слайди презентації в форматі Power Point (назва та мета роботи, аналіз існуючих інформаційних ризиків, можливий сценарій використання засобів моніторингу трафіку на рівні окремих пакетів, поширені засоби моніторингу мережевих пакетів, підготовчі операції для реалізації процесу сніфінгу, сніфінг пакетів засобами libpcap та їх наступний аналіз, висновки)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	24.10.23-26.10.23	виконано
2	Аналіз існуючих інформаційних ризиків	27.10.23-8.11.23	виконано
3	Огляд поширених засобів моніторингу мережевих пакетів	9.11.23-18.11.23	виконано
4	Підготовчі операції, необхідні для реалізації процесу сніфінгу	19.11.23-27.11.23	виконано
5	Сніфінг пакетів засобами libpcap та їх наступний аналіз	28.11.23-16.12.23	виконано
6	Висновки	17.12.23-19.12.23	виконано
7	Оформлення пояснювальної записки	20.12.23-5.1.24	виконано

Дата видачі завдання *24 жовтня* 2023 р.

Студент _____
(підпис)

Керівник роботи _____ *ст. викл. Твердохліб В.В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 60 с., 18 рис., 29 джерела, 3 додатки

BPF, IDS/IPS, APT, TDS, WIRESHARK, PCAP, TCPDUMP, АНАЛІЗ ПАКЕТІВ, МЕРЕЖЕВА БЕЗПЕКА

Об'єкт дослідження – інструменти дослідження трафіку мережі у середовищі Linux.

Мета роботи – дослідження існуючого інструментарію та методів побудови програмних засобів для аналізу мережевих пакетів.

Обгрунтовується роль засобів моніторингу трафіку у забезпеченні мережевої безпеки. Досліджуються програмні засоби аналізу пакетів на базі PCAP. Розглядається парадигма реалізації процесу сніфінгу та її базисні складові. Створюється програмний модуль для захоплення та подальшого аналізу пакетів у локальній мережі.

THE ABSTRACT

Explanatory note: 60 p., 18 fig., 29 sources, 3 apps.

BPF, IDS/IPS, APT, TDS, WIRESHARK, PCAP, TCPDUMP, PACKET ANALYSIS, NETWORK SECURITY

The object of the research is network traffic research tools in the Linux environment.

The purpose of the work is to study the existing tools and methods of building software tools for the analysis of network packets. The role of traffic monitoring tools in ensuring network security is substantiated. PCAP-based packet analysis software is being researched. The paradigm of the implementation of the sniffing process and its basic components are considered. A software module is created to capture and further analyze packets in the local network.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ

NGFW– (Next Generation Firewall) – міжмережевий екран наступного покоління;

IDS – (Intrusion Detection System) – система виявлення вторгнень;

IPS – (Intrusion Prevention System) – система протидії вторгненням;

TDS – (Threat Detection System) – система протидії загрозам;

APT – (Advanced Persistent Threat) – розвинута стійка загроза;

TCP – (Transmission Control Protocol) – транспортний протокол передавання даних;

UDP – (User Datagram Protocol) — протокол користувацьких дейтаграм, протокол негарантованої доставки;

IP – (Internet Protocol) – міжмережевий протокол, протокол мережевого рівня стеку TCP/IP;

BPF – (Berkeley Packet Filter) – фільтр пакетів Берклі;

NDIS – (Network Driver Interface Specification) — специфікація інтерфейсу мережевого драйверу, підсистема ОС.

.

ВСТУП

Розвиток інформаційних та комунікаційних технологій створює підґрунтя як для розробки та широкого впровадження широкого діапазону різнорідних мережевих послуг, так і удосконалення вже реалізованих.

При цьому, головним завданням таких сервісів є покращення тих чи інших аспектів повсякденної діяльності людини.

Водночас, синхронно з розвитком загального технологічного базису, має місце також розвиток механізмів та інструментів, які може бути використано зловмисником для:

- власного збагачення протиправними способами;
- зведення рахунків;
- нейтралізації конкурентів тощо.

У сьогоднішніх реаліях, а надто – кілька останніх років – інформаційний простір перетворюється на середовище для здійснення тих чи інших протиправних дій, зокрема, таких, як:

- викрадення інформації чи її модифікація інформації;
- псування, видалення чи викривлення критично важливої інформації;
- отримання доступу до інструментарію керування технологічними системами та платформами;
- реалізація афер у рамках фінансового сектору;
- розголошення інформації, що являє собою або конфіденційні дані, або комерційну таємницю, або грифовану таємницю державного рівня та ін.

У свою чергу, існування потенційних ризиків, базові категорії яких перелічено вище, зумовлюється тим фактом, що сьогодні практично усі сфери діяльності тим чи іншим чином присутні в онлайн-середовищі – або напряму, обо опосередковано.

Наприклад, інформаційна система компанії, організації чи установи так чи інакше може взаємодіяти з глобальним мережевим середовищем через:

- системи електронного документообміну які вона використовує;
- мережеві файлосховища – як класичні, так і на основі хмарних сервісів;
- системи електронних платежів.

Відтак – існують потенційні можливості для реалізації зловмисником теоретично будь-яких протиправних дій, можливих на базі мережевого середовища, оскільки суттєвий обсяг т.з. «сьогоднішніх» та/або майбутніх об'єктів інтересу онлайн є теоретично доступними.

Разом з тим, за кілька останніх років для зловмисників стала доступною велика кількість інструментів, за допомогою яких можуть бути реалізовані кібератаки як відносно одиничних хостів, так і стосовно окремих інформаційних систем підприємства, або взагалі усієї його мережевої інфраструктури. Такі інструменти, зокрема, надають майже усі хмарні платформи. Це, у першу чергу, стосується надання обчислювальних потужностей для побудови brootforce-атак для підбору паролів, а також анонімних проксі-серверів, використання яких суттєвим чином затруднює реалізацію комплексу дій з ідентифікації конкретних точок зловмисного втручання.

Водночас, окрім атак ззовні, не меншу небезпеку являють собою зловмисні дії, які можуть виконуватися усередині мережі, як наслідок:

- впливу т.з. «людського фактору»;
- некваліфікованих дій користувачів;
- цілеспрямованих дій користувачів мережі.

При цьому, ознаки існування потенційних загроз даної категорію у звичайному режимі частіше за все майже неможливо виявити до моменту їх реалізації. Це зумовлено тим фактом, що більшість традиційних засобів кіберзахисту, як то мережеві екрани, NGFW та антивірусні системи або орієнтовані виявлення аномалій на рівні периметру мережі, або орієнтуються на пошук характерних ознак зловмисної присутності, відомих тому чи іншому засобові захисту. Це саме також є справедливим для IDS/IPS систем.

У таких умовах моніторинг трафіку, яких циркулює усередині мережі та передається назовні, додатково дозволяє виявити ті чи інші поведінкові аномалії окремих вузлів, які, у свою чергу, з одного боку можуть не суперечити правилам IDS/IPS-засобів, але при цьому виконувати:

- обмін даними з «сумнівними», або апріорі небезпечними вузлами;
- обхід правил регламенту функціонування мережі, встановлених алдміністратором, або фахівцем з інформаційної безпеки.

При цьому, виявлення та далі – блокування подібних поведінкових аномалій суттєвим чином здатне позитивно вплинути на рівень інформаційної безпеки мережі у цілому, та стати на заваді проведення

найбільш небезпечних зловмисних атак на мережеву інфраструктуру – APT (розвинута стійка загроза).

Таким чином, дослідження, розробка та впровадження інструментів моніторингу трафіку на сьогодні являє собою надзвичайно гостре та актуальне науково-прикладне завдання, що і зумовлює актуальність даної кваліфікаційної роботи.

1. АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ РИЗИКІВ

1.1 Огляд ключових об’єктів кібератак та їхніх можливих наслідків

Відповідно до матеріалів дослідження [1] на поточний момент ключовими цілями зловмисного впливу є (рис.1.1):

- підприємства фінансового сектору (банки та фінансові організації у цілому);
- підприємства промисловості, а також об’єкти критичної інфраструктури;
- структури державного сектору.

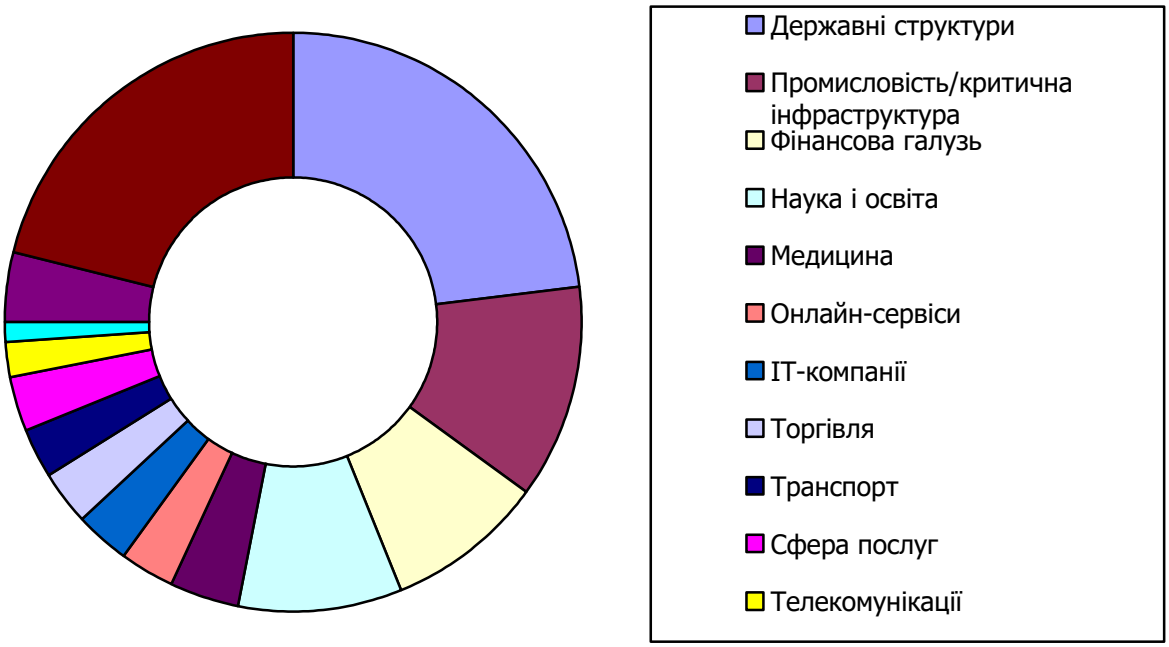


Рисунок 1.1 – Найчастіші об’єкти кібератак

На той випадок, коли ті чи інші атаки будуть вдалими, їхні наслідки відразу, чи згодом, будуть мати критичні наслідки на рівні регіону, окремих економічних галузей, частини суспільства або усієї держави. Такі наслідки можуть мати прояв у вигляді:

- суттєвих фінансових виратах;
- втратах репутаційного типу;

- псування мереж та інформаційних систем загальнодержавного чи транснаціонального рівнів та виникнення негативних наслідків у коротко- чи довгостроковій перспективі;
- суттєвих ризиків реалізації глобальних чи регіональних техногенних катастроф.

Разом з тим слід зазначити, що у суттєво меншому масштабі з позиції можливої загрози для суспільства чи його частини у цілому, але не менш небезпечними, розглядаються кібератаки дешо іншої орієнтованості. Мається на увазі атаки, які є орієнтованими на інформаційну інфраструктуру структур комерційного сектору, що частіше за все є наслідком таких чинників, як:

- недобросовісна конкуренція;
- зведення рахунків з керінийтвом колишніми співробітниками;
- особисте збагачення;
- використання інформаційної інфраструктури суб'єктів комерційної діяльності як проміжних ланок для подальшого розповсюдження атаки тощо.

На той випадок, коли кібератаки на мережеву інфраструктуру комерційного сектору є вдалими, їх реалізація у будь-якому випадку веде до виникнення фінансових збитків та інших втрат. Деталізація результатів кібератак для даного випадку наводиться табл.1.1.

Таблиця 1.1 – Типові ймовірні наслідки реалізованих кібератак на рівні комерційного сектору

Результат атаки	Наслідок
Компрометація даних ключових персон	Репутаційні втрати
Втрата критичних даних	Втрата конкурентних переваг
Крадіжка комерційної таємниці	Втрата довіри клієнтів
Пошкодження IT-інфраструктури	
Призупинення бізнес-процесів	Зменшення частки, яку компанія займає на ринку
Недоступність сервісів, що надаються	Прямі і опосередковані фінансові втрати
Крадіжка фінансових активів	

При цьому, з початку 2011 року обсяг вдалих кібератак, які було щоденно реалізовано на території Європейського Союзу, перевищує позначку у 100.

Разом з тим, обов'язково слід зазначити, що найбільшу як перспективну, так і реальну загрозу для будь-якої інформаційної системи складає APT.

Зокрема, розвинута стійка загроза, як загроза критичного типу на державному та транснаціональному рівнях, результативні APT періодично реалізуються з першої половини 2000-х років.

1.1.1 Найбільш результативні кібератаки, реалізовані за останні кілька років у рамках ідеології APT

На сьогодні найбільш відомими, масштабними та успішними серед APT-кібератак вважаються:

- комплексні атаки на мережеву інфраструктуру NASA та пентагону (2000-й рік);
- вдале втручання у роботу інформаційних систем Los Alamos National Laboratory та Oak Ridge National Laboratory у 2007 році;
- неодноразовий злам інформаційної системи, що належала міністерству оборони США протягом 2008 року;
- зловмисний вплив Operation Aurora, у результаті якого було досягнено входу за периметр внутрішньої мережі Google і викраденням первинних кодів переліку продуктів компанії протягом 2009 року;
- результативні злами мережевої інфраструктури урядів Австралії, Канади та Франції, у наслідок чого сторонні особи отримали у розпорядження велику кількість документів з таємного листування (період 2010-2011 років);
- злам мережі МВФ (період 2011-2012 років).

Водночас, реальний перелік успішних масштабних атак з 2000 року і понині є значно ширшим – це пояснюється тим, тому що розголосу набули лише найбільш гучні факти.

При цьому, слід зазначити, що представниками як профільних організацій та відомств національних рівнів, так і керівниками структур, задіяних у галузі інформаційної безпеки, вироблено ряд протоколів, які

перешкоджають розголошенню інформації про випадки зламу великих інформаційних систем.

1.2 Аналіз механізмів реалізації APT-атак

1.2.1 Загальна специфіка перебігу APT

Беручи до уваги перелік успішних атак, наведений у п.1.1.1, можна відмітити наступне:

- очевидно, що кожна з інформаційних систем, наведених у переліку, була оснащена потужними засобами виявлення та протидії кіберзагрозам;
- питаннями забезпечення кіберзахисту у рамках кожної з мереж, що зазнала зламу, займався на постійній основі штат висококваліфікованих профільних фахівців;
- зловмисниками було не лише подолано переметр мережі, але також отримано доступ до критично важливої інформації; при цьому, зловмисна присутність зберігалася протягом достатньо довгого часу (порядку 10-12 місяців) без її виявлення.

Відтак, вищезазначене вказує на:

- високий рівень результативності APT;
- неможливість протидії APT типовими організаційно-технічними засобами навіть за умови найвищого рівня технологічної оснащеності та професійної підготовки фахівців з кібербезпеки;
- низьку ймовірність виявлення зловмисної присутності усередині мережі.

Подібна результативність зумовлюється самою архітектурою APT, що вибудовується за унікальним сценарієм, який орієнтований на конкретну інформаційну систему.

Таким чином, атака APT не має глобально-типових ознак, які може бути використано для виявлення її присутності.

Разом з тим, APT передбачає виконання таких типових стадій реалізації, як (рис.1.2):

- передуюча стадія (стадія попереднього планування);
- стадія подолання периметру (проникнення);
- стадія розповсюдження впливу присутності;
- стадія досягнення мети.

При цьому, на передуючій стадії зловмисник, у сутності, вивчає майбутній об’єкт атаки, збираючи дані стосовно:

- архітектури цільової інформаційної системи;
- існуючих мережевих регламентів;
- застосовуваних систем протидії кіберзагрозам;
- штатного складу персоналу, який має ті чи інші привілеї усередині мережі тощо.

Дана стадія, у тому числі, передбачає збір раніше перелічених відомостей від контрагентів, діючих чи раніше звільнених співробітників а також з відкритих джерел.

Результатом передуючої сталії є ряд ймовірних сценаріїв проникнення, а також розробка чи збір технологічних інструментів, на базі яких сценарії може бути реалізовано на наступних стадіях.

Рисунок 1.2 – Стадійність реалізації загального сценарію APT

Разом з тим, на стадії проникнення усі завдання, які потребують рішення, зводяться до обходу захисних засобів периметру, з подальшим розміщенням зловмисних модулів.

Типовий набір модулів, які при цьому може бути застосовано, фактично відсутній, так як він визначається з огляду на конкретні умови. Проте, у загальному випадку, стадія проникнення передбачає застосування ряду підходів та методів та підходів, серед яких з високою ймовірністю можуть бути:

- інструменти соціальної інженерії (окремим рядком - методи фітингу у різних його варіаціях);
- використання т.з. загроз нульового дня;
- атаки, орієнтовані на раніше виявлені уразливості конкретного прикладного ПЗ або його окремих модулів тощо.

Далі, протягом стадії розповсюдження, реалізуються технологічні кроки з поступового інфікування ключових складових мережевої інфраструктури зловмисним кодом.

Наслідком даних дій є отримання зловмисником доступу до окремих пристроїв чи різного рівня систем мережі, або цільових мережевих ресурсів узагалі.

У свою чергу, стадія досягнення мети розвинутої стійкої загрози, у порівнянні з будь-якими іншими стадіями її реалізації, може тривати суттєво більший проміжок часу.

Така особливість перебігу даної стадії зумовлена тим фактом, що зловмисник, досягши поставленої першочергової мети, тим паче зберігати присутність усередині мережі.

За таких умов йому є доступними дані, що зберігаються у локальних та мережевих файлосховищах, та доступною є можливість прямо чи опосередковано чинити вплив на усі бізнес-процеси атакованої компанії.

1.2.2 Засоби реалізації APT

Ще одна характерна особливість APT – широкий діапазон засобів реалізації, так і шляхів, якими дані засоби спрямовуються усередину переметру атакованої цільової мережі.

Перелік шляхів, якими зловмисні агенти найчастіше потрапляють до цільових мереж, показано рис. 1.3.

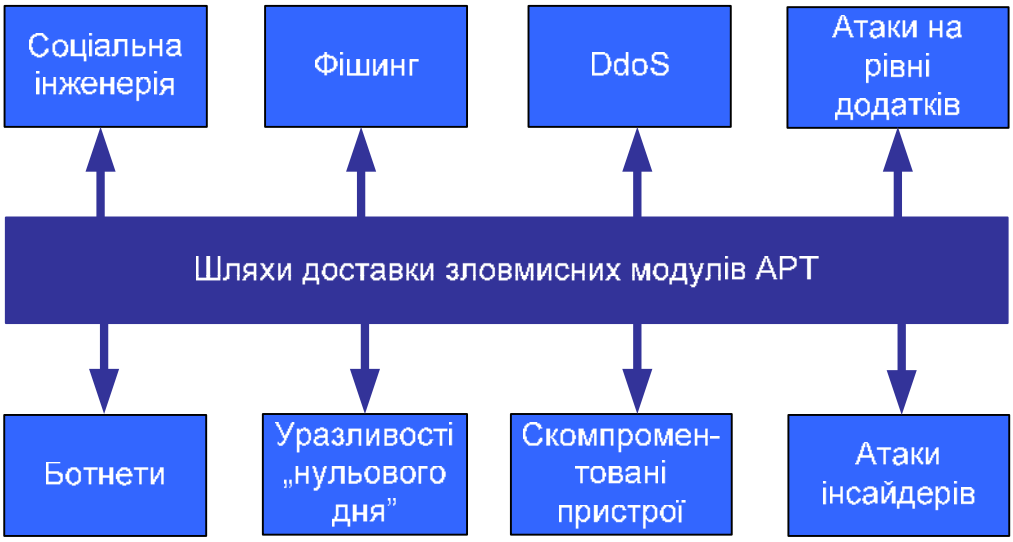


Рисунок 1.3 – Шляхи, якими зловмисні агенти потрапляють усередину параметру мережі

При цьому, якщо говорити про засоби реалізації APT, слід зазначити, що їх перелік, як раніше зазначалося є широким, та також вилика кількість засобів використовується спільно.

Найчастіше це:

- різноманітні exploit (експлойти);
- валідатори;
- даунлоадери (завантажувачі);
- дроппери (dropper);
- базові, або ключові функціональні модулі (payload).

Разом з тим, сценарії функціонування перелічених засобів за великим рахунком подібні у межах кожного з зазначених класів та утілюються на принципах, схожих у цілому.

Водночас, винятком тут є модулі payload, що створюються або як ізольовані самодостатні модулі, або як упорядкована множина окремих команд або інструментів, орієнтованих на рішення конкретних завдань, з урахуванням відомостей, які було зібрано на передуючій стадії.

Проте важливим є те, що перелік вищезазначених засобів не є фіксованим.

Нерідко, беручи до уваги конкретні умови, його може бути скореговано - зокрема, шляхом виключення з нього ряду складових.

Це, частіше за все, можуть бути або дроппери або завантажувачі.

У свою чергу, вищезазначені засоби реалізації APT завантажуються до атакованої мережі різними каналами надходження (таб 1.2).

Таблиця 1.2 – Канали,що найчастіше задіюються для доставки засобів APT

Клас засобів	Канал доставки
Exploit	E-mail, USB-пристрої, веб-сайти
Валідатор	E-mail-вкладення, фішингові сайти
Downloader	E-mail-вкладення, фішингові сайти
Dropper	E-mail, веб-сайти, Exploit, валідатори
Payload	Dropper, Downloader, Exploit, валідатори, E-mail, веб-сайти

Отночасно з цим, слід звернути особливу увагу на особливу роль як соціальної інженерії у цілому, так і фішингу, як її специфічного підвиду, у реалізації атак APT.

Перш за все, слід зазначити, що:

- вплив людського фактору, як базису реалізації практично будь-якої кібератаки, виключити практично неможливо;
- будь-яка мережа, що потенційно може бути об’єктом інтересу зловмисників, є потенційно вразливою, як наслідок впливу людського фактору;
- розгляд людського фактору, як одного з чинників, що може мати вплив на результативність APT, є нетривіальним завданням, рішення якого може мати лише часткові рішення.

1.3 Роль соціальної інженерії та людського фактору узагалі у реалізації розвинутих стійких загроз

Інформаційна безпека будь-якої мережі, що має у своєму складі хоча б елементарні компоненти кіберзахисту, залежить від:

- рівня зацікавленості потенційного зловмисника щодо зламу мережі, щорозглядається;
- інструментарію, який застосовано для протидії кібератакам (мережеві екрани, IDS та IPS-системи, NGFW, TDS, антивірусні комплекси тощо) та його можливостей;

- професіоналізма фахівців з інформаційної безпеки, які обслуговують мережу, та їх наявності взагалі;
- внутрішнього мережевого регламенту;
- обізнаності персоналу з найбільш поширених питань та проблематики у сфері інформаційної безпеки.

При цьому, слід зазначити, що мережа підприємства зазнає підвищених ризиків у випадках, коли:

- користувач має змогу без обмежень паралельно вести обмін повідомленнями та листування з зовнішніми абонентами з використанням E-mail та облікових записів месенджерів як наданих підприємством, так і персональних;
- регламент функціонування внутрішньої мережі не передбачає блокування усіх інших облікових записів E-mail та месенджерів, окрім професійних.

Разом з тим, навіть у випадку, коли внутрішній регламент безпеки не дозволяє співробітнику користуватися web-версіями месенджерів у рамках персональних акаунтів (це також справедливо для персональних E-mail), надзвичайно важко, а деколи - фактично неможливо блокувати можливість використання користувачами корпоративних облікових записів для здійснення:

- реєстрації на стороннії мережевих сервісах «за інтересами»;
- паралельного використання корпоративних месенджерів та E-mail у приватних цілях.

Тим самим створюються умови для можливості реалізації методів соціальної інженерії у межах захищених корпоративних мереж незалежнол від їх архітектури та особливостей.

ЗА може і не використовуватися.

Щодо соціальної інженерії - класичним її прикладом може вважатися фішинг (*англ*. phishing від fishing «рибальство, вивужування»).

Фішинг на сьогодні – один з найбільш розповсюджений видів інтернет-шахрайства, спрямований на отримання логіків та паролів користувачів, та/або інших конфіденційних даних.

В основі фішингу лежить реалізація процедиру автоматичного розсилання електронних листів широкому діапазону випадкових користувачів. Існує також варіант фішингу, коли зловмисник виконує

розсилання листів за певним переліком адрес, до яких він попередньо отимав доступ.

Зазвичай це можуть бути листи від імені популярних брендів, особисті листи та повідомлення усередині різноманітних мережевих сервісів. Зокрема, це можуть бути листи від імені того чи іншого банку, або адміністрації соціальних мереж.

Такі листи нерідко міститься пряме посилання на сайт, яке не має зовнішніх відмінностей від реального відправника, або на сайт, де міститься редирект.

У підсумку, після того, як користувач здійснить перехід до підробленої сторінки, зловмисники далі частіше за все застосовують різноманітні психологічні прийоми для того, метою яких є спонукання користувача до введення на даній сторінці (що не є справжньою, хоча зовні ідентичною) власний логін та пароль, що використовуються ним для доступу до того чи іншого сайту.

Таким чином, це дає змогу зловмисникам отримати доступ до облікових записів та банківських рахунків.

Фішинг базується на незнанні (або ігноруванні) користувачами основ мережевої безпеки.

Зараз для захисту від фішингу розробникип більшості браузерів за домовленістю використовують схожі (або однакові) способи інформування користувачів про те, відкритий ними сайт є потенційно небезпечним. Також зараз оновлені версії багатьох браузерів мають у своєму складі засоби захисту від фішингу.

1.4 Використання зловмисниками агентурного ресурсу

Припустмо, що один, або декілька співробітників деякого підприємства, що є одним з об’єктів інтересу зловмисника, беруть участь у реалізації APT. За даних умов кожен з них розглядається як зловмисний агент (ЗА).

У даному разі їхня участь можлива щонайменше у наступних форматах:

- ЗА отримує від ініціатора зловмисного впливу файли, що містять зловмисні модулі (або зловмисні модулі у чистому вигляді – наприклад,

функціональні частини коду), а також інструкції з їх розміщення, та надалі розміщує їх у потрібну локацію;

- ЗА отримує від ініціатора зловмисного впливу посилання у корпоративній/власній пошті разом з рекомендаціями, на які з них необхідно відреагувати, перейшовши за зазначеними лінками (даний перелік дій, у сутності, аналогічний тому, що має місце у ході реалізації процедури фішингу).

У будь-якому випадку з зазначених, у результаті виконаних з боку ЗА дій, усередині периметру мережі спостерігатиметься процес, який може включати у себе, зокрема:

- отримання зловмисником доступу до одного чи ряду критичних вузлів/сервісів, що беруть участь у забезпеченні захищеності мережевої інфраструктури;
- пошук даних того чи іншого роду, що являють собою інтерес для зловмисника;
- передавання знайдених даних до тих чи інших вузлів, доступ до яких має зловмисник;
- «м’яку» зміну регламенту функціонування мережі та режиму захисту тощо.

1.5 Обгрунтування актуальності використання моніторингу трафіку для виявлення ознак зловмисного втручання

Частіше за все, незалежно від конкретного сценарію зловмисного впливу, який реалізується, кожна з його реалізацій матиме ряд схожих рис, а саме:

- управління зловмисними модулями, їх збірками та усією атакою взагалі (як процес обміну пакетами-командами та пакетами-підтвердженнями) здійснюється у загальному випадку з довільних зовнішніх вузлів;
- передавання даних (якщо атака зводиться до крадіжки інформації) виконується до вузлів або тих самих, з яких надходять керуючі команди, або інших вузлів ззовні.

Таким чином, у загальному випадку опосередкованою ознакою ймовірної атаки може бути обмін даними внутрішніх вузлів мережі з вузлами зовнішніми.

У загальному випадку, якщо мережа не має встановленого жорсткого регламенту функціонування, кожен з клієнтських вузлів по відношенню до зовнішнього оточенням має можливість вести обмін даними без жодних обмежень.

Разом з тим, щонайменше у випадках, коли безпеці на рівні корпоративної мережі приділяється особлива увага, регламент її функціонування передбачає блокування вхідного, вихідного або будь-якого трафіку взагалі з:

- вузлами, що марковані пошуковими системами, як потенційно зловмисні;
- вузлами з сумнівною репутацією;
- вузлами, що розглядаються як потенційно небезпечні, виходячи з внутрішнього регламенту безпеки конкретної компанії чи відомства;
- будь-якими вузлами, що не входять у т.з. «довірений список».

Виходячи з вищезазначеного, можемо дійти до таких проміжних висновків:

- факти обходу мережевого регламенту шляхом обміну даними з забороненими вузлами зазвичай напряму можуть свідчити про наявність кібератаки;
- для виявлення опосередкованих ознак кібератаки необхідно мати відомості відносно того, якими даними, та з якими вузлами ззовні, здійснюють обмін вузли усередені мережі;
- за умови «м’якої» зміну регламенту функціонування мережі та режиму її захисту не гарантується, що IDS/IPS засоби матимуть змогу виявлення аномалій функціонування мережевого середовища усередині периметру.

Таким чином, застосування засобів моніторингу трафіку на рівні пакетів, як додаткового інструментарію виявлення кіберзагроз, дає можливість виявлення зловмисних втручань з набагато більшим рівнем ймовірності, ніж за умов, коли у системі використовується стандартизований набору засобів кіберзахисту.

Один з можливих сценаріїв використання засобів моніторингу трафіку на рівні окремих пакетів для підвищення безпеки мережевої інфраструктури показано рис.1.4.

Згідно даного сценарію, протягом його реалізації виконуються наступні операції:

1. Прийом пакету даних.
2. Зчитування адреси джерела пакету.
3. Локалізація джерела (перевірка – зовнішній чи внутрішній мережі належить пакет).
4. Перевірка привілеїв хоста (на той випадок, коли локальний хост надсилає пакети зовнішньому вузлу).
5. Перевірка записів стосовно джерела пакету, якщо його надіслано з зовнішньої мережі.
6. Перевірка налаштувань безпеки у випадках, якщо локальний хост – джерело пакетів – може їх передавати назовні у рамках чинної політики, або – якщо зовнішнє джерело пакетів не внесено до переліку заборонених ресурсів.
7. Надання можливості обміну даними локальному хосту та/або зовнішньому вузлу, якщо налаштування безпеки не зазнало змін, а відповідні локальне та зовнішнє джерела мають приводи для обміну даними.
8. Повне блокування каналу, якщо буде виявлено, що справедливим є хоча б одне з наступних тверджень:

- локальний хост, який робить спробу обміну даними з тим чи іншим вузлом ззовні, не має відповідних привілеїв згідно існуючої схеми безпеки;
- зовнішнє джерело, з якого отримуються пакети або не належить до переліку дозволених, або належить то т.з. «чорного списку» - тобто, потенційно небезпечних вузлів.

Тобто, якщо існуюча політика мережевої безпеки передбачає існування списків дозволених зовнішніх вузлів, та вузлів усередині мережі, які мають привілеї з обміну даними з зовнішніми приймачами, при цьому, у результаті виконаного моніторингу пакетів виявлено, що фактична ситуація не відповідає базовим налаштуванням політики, слід вважати, що усередині мережі реалізовано атаку, у ході якої, зокрема внесено модифікації опцій у засоби безпеки.

Отже, актуальним є дослідження та впровадження інструментарію, що дозволяє здійснювати аналіз пакетів, які передаються мережею.

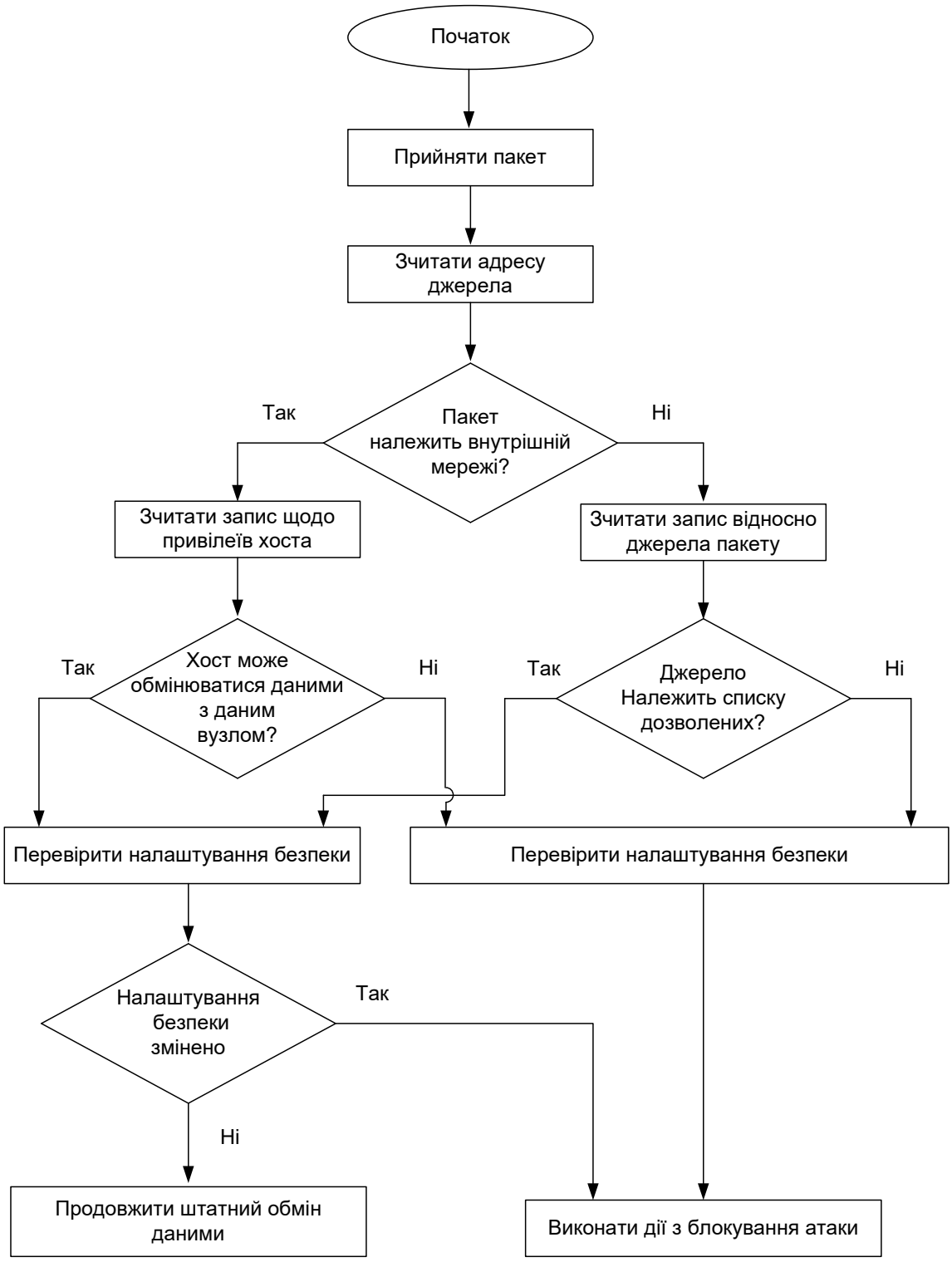


Рисунок 1.4 - Один з можливих сценаріїв використання засобів моніторингу трафіку на рівні окремих пакетів

2. ОГЛЯД ПОШИРЕНИХ ЗАСОБІВ МОНІТОРИНГУ МЕРЕЖЕВИХ ПАКЕТІВ

2.1 Базові класи зособів моніторингу пакетів

У загальному випадку, уся множина інструментів, які може бути використано для перегляду мережевих пакетів у мережах Windows, належить одному з класів засобів, а саме:

- сніфери;
- засоби IDS/IPS.

При цьому, одним з найчастіше використовуваним сніферів, щодає змогу дослідження пакетів у мережах Windows та мережах Unix-сімейства, є WireShark.

У свою чергу, як приклад засобу IDS/IPS для Windows та Unix-мереж доцільно розглянути програмний пакет Surricata як такий, що характеризується досить широкою аудиторією користувачів.

Виконаємо дослідження кожного з засобів окремо.

2.2 Програмний засіб WireShark

В основі роботи як WireShark, так і сніферів узагалі, є інструментарій прослуховування подій зарального розподіленого середовища на рівні мережевого адаптеру клієнтського вузла.

При цьому, може прослуховуватися як один мережевий інтерфейс (наприклад, активний за замовчуванням, або активний у поточний час), так і ряд інтерфейсів одночасно. У т.ч. останні релізи WireShark надають можливість отримувати та аналізувати пакети також з безпровідних інтерфейсів ряду типів (рис. 2.1.).

Разом з тим, у штатному режимі роботи, будь-який мережевий адаптер, хоча має фізичну можливість прийому, у сутності, будь-якого пакету, що надсилається загальним мережевим середовищем, обмежений прийомом лише тих пакетів, у полі призначення яких вказано його адресу.

Виходячи з зазначеного вище, для початку процесу снiфінгу, тобто, перехоплення усієї множини пакетів, мережевий адаптер переводиться у т.з. *нерозбірливий режим* (promisque mode).

Далі, керуючись чинними опціями, може виконуватися захоплення пакетів з мережі з подальшим аналізом. Засіб надає досишь широкий інструментарій конфігурування режимів роботи.

Стосовно режимів захоплення пакетів у рамках WireShark множина опцій має назву фільтра.

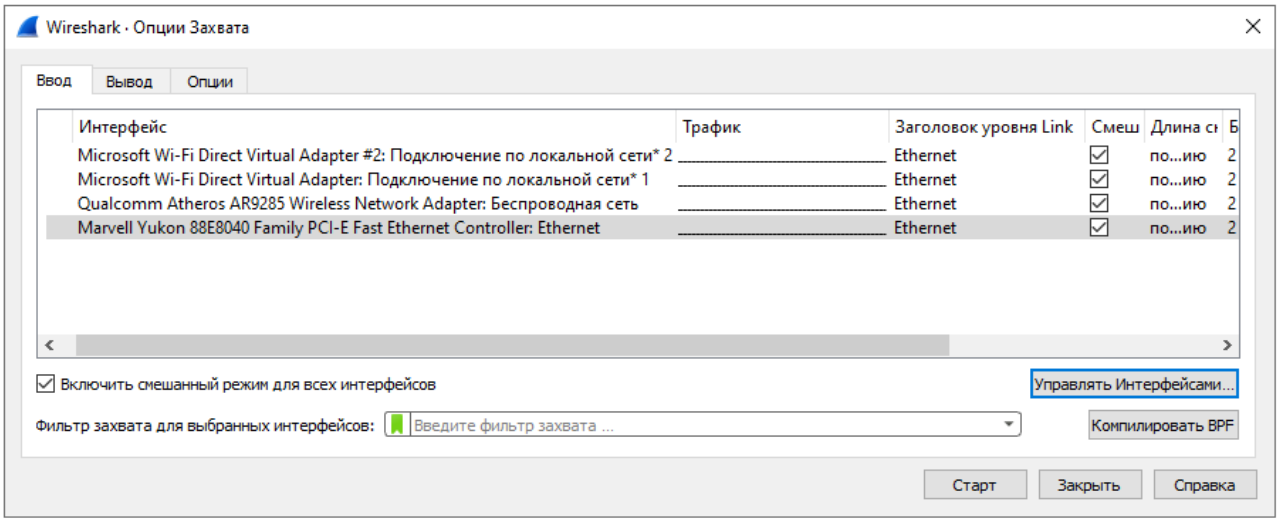
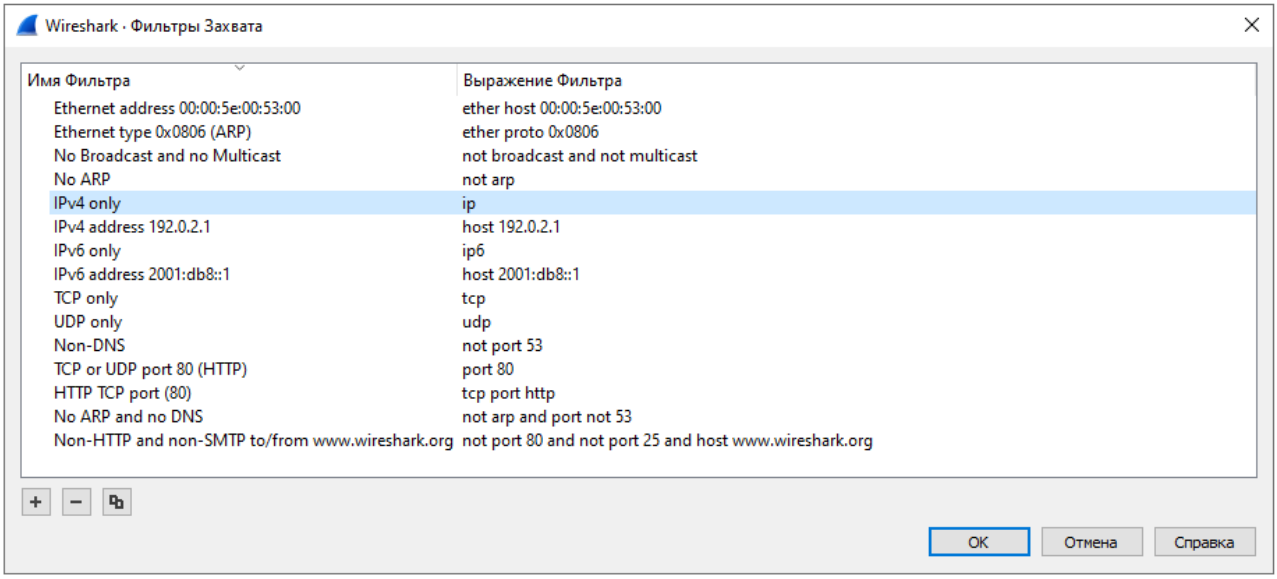


Рисунок 2.1 – Вікно налаштування інтерфейсів WireShark

При цьому, передбачено можливість використання фільтрів за замовчуванням (рис.2.2).



Рисвунок 2.2 – Приклад набору фільтрів за замовчуванням

За рахунок можливості гнучкого конфігурування фільтрів, можна захоплювати для наступного аналізу лише ті пакети, що першочергово потребують уваги.

Наприклад, це пакети, що відповідають одній або кільком умовам:

- безпосередньо беруть участь у передаванні даних (TCP, UDP, IP та ін.);
- належать внутрішнім вузлам з тими чи іншими рівнями привілеїв;
- надходять з зовнішніх вузлів, які не внесено до переліку дозволених (рис. 2.3) тощо.

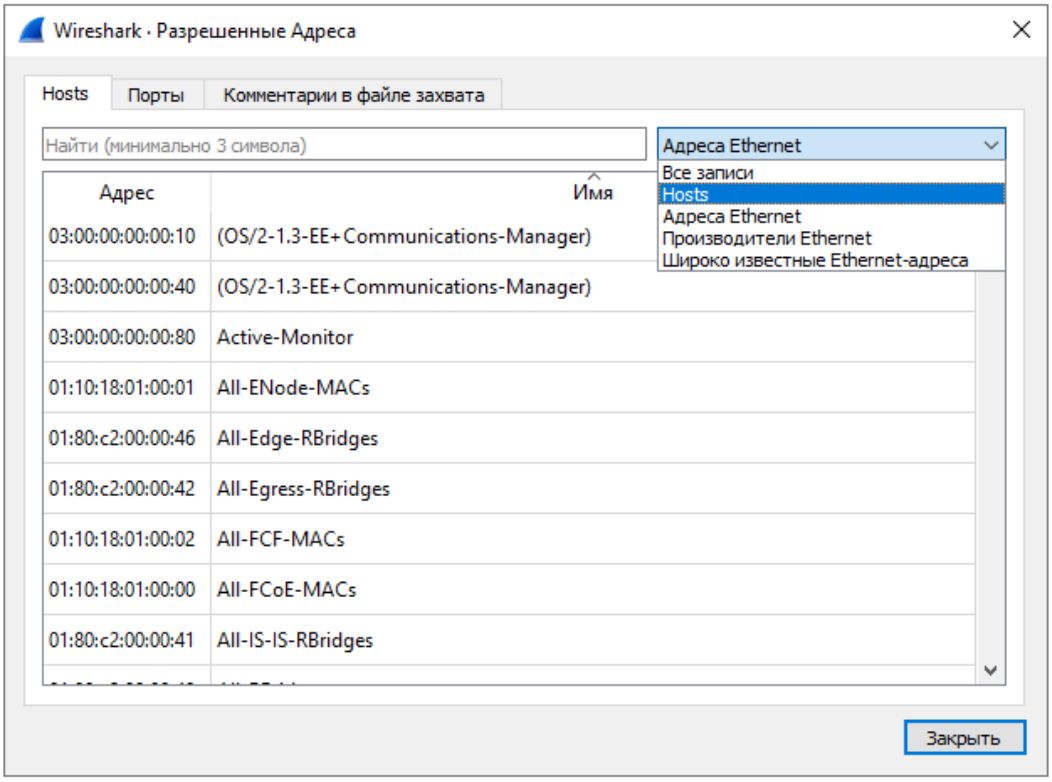


Рисунок 2.3 – Інтерфейс налаштування переліку дозволених адрес WireShark

Таким чином, можна аналізувати лише ті з пакетів, що можуть належати підозрілим сесіям інформаційного обміну.

У свою чергу, захоплені пакети виводяться на перегляд у розширеному форматі.

Так, відповідний інтерфейс програмного засобу містить, як показано на рисунку 2.4:

- перелік пакетів, отриманих під час поточного сеансу захоплення;

- вікно деталізації відомостей про захоплений пакет;

- двійковий зміст пакету, який надається для перегляду у HEX-форматі.

При цьому, для кожного пакету фіксуються відомості про:

- час надходження;

- адреси джерела та отримувача;

- протокол;

- додаткові дані, які об’єм яких залежить від належності пакету тому чи іншому типу.

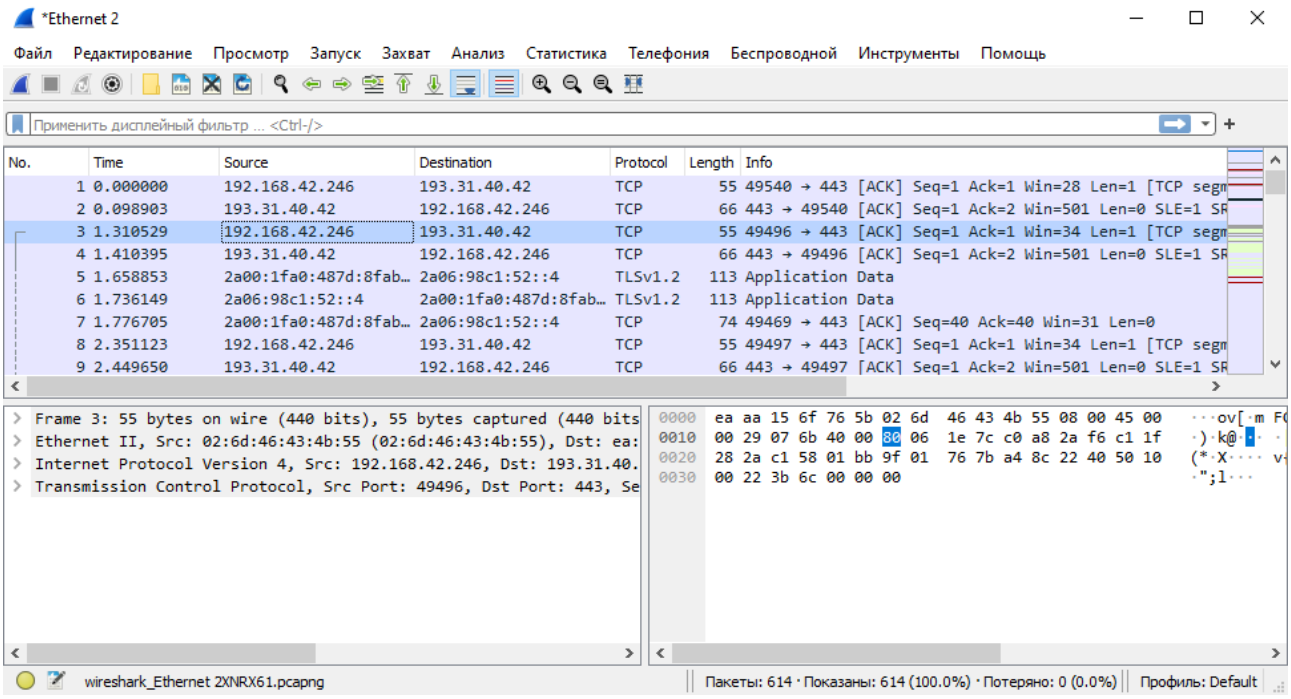


Рисунок 2.4 – Перегляд відомостей про один з захоплених пакетів

Як видно з рис. 2.4, даний пакет було спрямовано з вузла 192.168.42.246 до вузла з адресою 193.31.40.42. При цьому, у якості транспотру використано TCP-протокол.

Далі, у лівому нижньому вікні бачимо, що доступним для перегляду є уточнений перелік даних, зведений у 4 групи.

Так, на рис. 2.5 наводиться приклад розгорнутої інформації стосовно TCP-протоколу для нашого випадку.

Зокрема, звідси можна отримати інформацію про порт призначення та порт джерела, контрольну суму пакету, величину користого навантаженна та ін.

Таким чином, згідно парадигмі, зображеній на схемі, поданій рис.1.4, засіб WireShark дозволяє переглядати фактичну ситуацію стосовно обміну пакетами даних мережевих вузлів та вузлів ззовні мережевого периметру.

При цьому, якщо згідно з існуючою політикою безпеки попередньо було встановлено обмеження на перебіг даного процесу, у той же час у ході аналізу пакетів виявлено, наступне:

- здійснюється передавання даних мережевими хостами до вузлів, які належать до списку заборонених;
- мережеві вузли отримують пакети з попередньо заборонених хостів;

відтак – хоча б в одному випадку з зазначених є підстава вважати, що:

- на даний момент мережева інфраструктура підлягає зловмисному впливу (ознакою якого є обхід чинної політики);
- політика безпеки була налаштована некоректно.

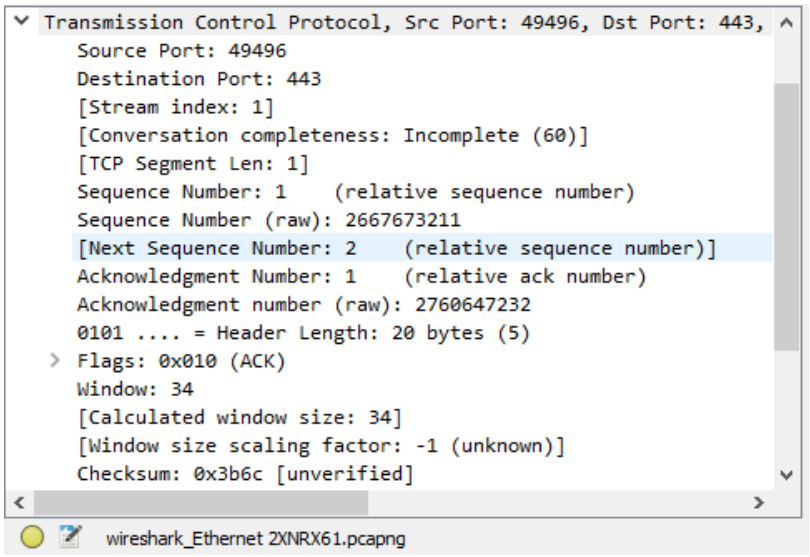


Рисунок 2.5 – Інтерфейс отримання додаткових відомостей про перехоплений пакет

У будь-якому випадку, виявлення невідповідності встановлених опцій безпеки з існуючим режимом функціонування мережі та/обо окремих пристроїв у її складі є вагомим аргументом на користь необхідності проведення додаткової перевірки налаштування усих програмно-апаратних модулів, задіяних для забезпечення безпеки.

У свою чергу, ряд механізмів, реалізованих у рамках WireShark, використовує засіб Suricata.

2.3 Програмний засіб Suricata

На відміну від Wire Shark, що являє собою сніфер у чистому вигляді, Suricata використовує сніфінг, як один з інструментів комплексної IDS/IPS-моделі.

Засіб Suricata, при цьому, є хост-орієнтованим засобом, тобто, функціонує на рівні окремих вузлів, та не потребує використання серверної компоненти.

При цьому, програмний пакет дозволяє гнучко налаштовувати правила безпеки.

Можливим є як використання універсальних правил, які завантажуються з офіційного сайту проекту, так і:

- створення власних правил;
- модифікація правил зі стандартизованого переліку, згідно власних потреб;
- вибір конфігурації правил, актуальної для тієї чи іншої ситуації; тут деактивується деякий перелік правил, що не є цінними, та залишаються активними ті з них, використання яких матиме сенс у даній конфігурації мережі та ситуативно.

Стосовно сніфінгу, Suricata не надає аналітичного інструментарію для дослідження пакетів подібного тому, який присутній у складі WireShark.

При цьому, дослідження пакетів здійснюється в автоматичному режимі. Кожен перехоплений пакет, та вузол, що його породжує/прийнає зазнають перевірки на відповідність внутрішнім правилам безпеки. Зокрема, перевіряються:

- належність адрес джерела та призначення пакету до санкційних списків;
- привілеї вузла та надані повноваження щодо можливостей взаємодії з тими чи іншими зовнішніми/внутрішніми мережевими ресурсами;
- привілеї вузла відносно можливості використання тих чи інших мережевих додатків та/або сервісів в існуючій ролі;
- відповідність вузла ролі, яку було призначено адміністратором.

Очевидно, що безумовною перевагою автоматичного аналізу пакетів є:

- по-перше, висока продуктивність процесу дослідження та прийняття рішення щодо присутності ознак зловмисного втручання;

- по-друге, відсутність необхідності для адміністратора чи фахівця з безпеки власноруч вести моніторинг, хоча б на рівні випадкових вибірок з загального масиву захоплених пакетів.

Разом з тим, недоліком підходу до аналізу пакетів, реалізованого у рамках як Suricata, так і подібних IDS/IPS систем, є необхідність чіткого конфігурування правил безпеки з урахуванням:

- актуального мережевого регламенту;

- апаратно-програмної конфігурації мережі;

- специфічних характеристик мережі, у т.ч. потенційно можливі загрози та ймовірні об’єкти атаки.

При цьому, у базовій конфігурації системи початково існує надмірна кількість правил безпеки (рис.2.6).

Зрозуміло, що з позиції забезпечення балансу між захищеністю системи та швидкодією, доцільним є створення власної конфігурації правил, та, у ряді випадків, внесення змін у той чи інший перелік стандартизованих правил. Відтак, за зазначених умов процес налаштування засобу Suricata може потребувати суттєвих часових витрат.

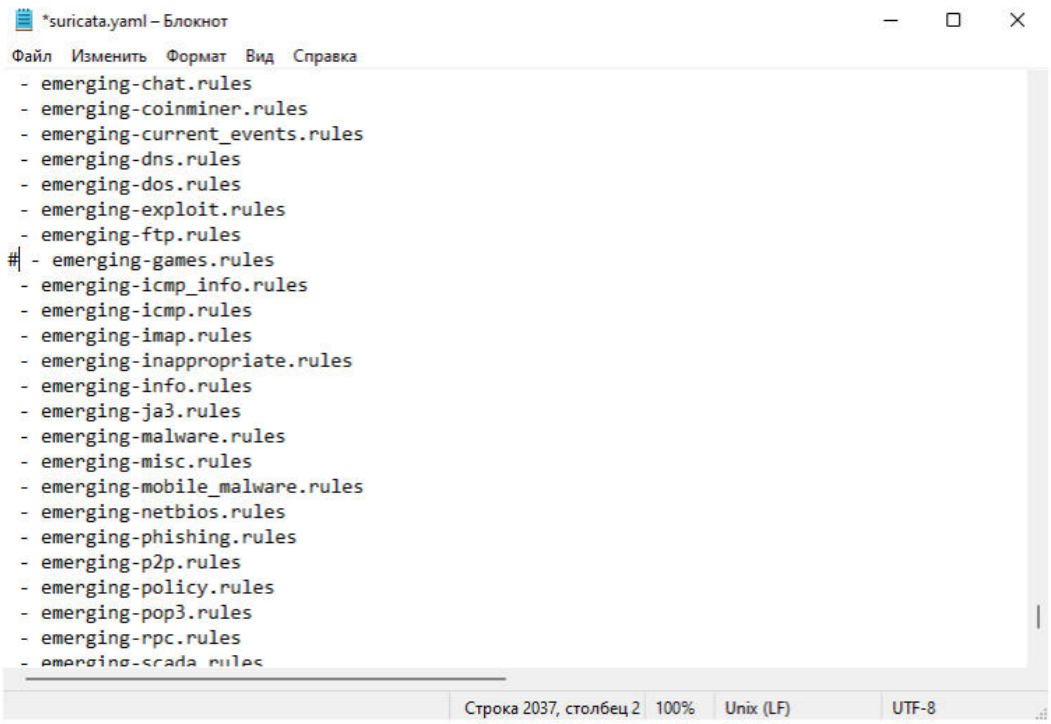


Рисунок 2.6 – Налаштування переліку активних правил yaml-файлу Surricata

У свою чергу, у рамках механізмів аналізу пакетів, Surricata функціонує згідно з алгоритмом, наведеним рис. 1.4.

2.4 Утиліта TCPDump

На відміну від вищезазначених засобів, TCPDump першочергово створено для аналізу мережевого трафіку у Unix-середовищі як проект з відкритим кодом. Пізніше на базі даного проекту було реалізовано:

- WinDump – також проект з відкритим кодом, орієнтований на Windows-середовище;
- TCPDump for Win – комерційне ПЗ з аналогічним функціоналом, але простіше у використанні.

Класична утиліта TCPDump – повінстю консольний засіб (рис.2.7). При цьому, формат виводу даних у загальних рисах аналогічний головному вікну WireShark.

```
12:23:12.857291 IP 162.159.130.234.https > vulp-nezuko.38732: Flags [P.], seq 296367:296427, ack 794, win 39, length 60
12:23:12.857295 IP vulp-nezuko.38732 > 162.159.130.234.https: Flags [.], ack 296427, win 9902, length 0
12:23:12.858079 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4181199:4182595, ack 15771, win 379, options [nop,nop,TS val 758433630 ec
r 218847144], length 1396
12:23:12.858102 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4182595, win 4328, options [nop,nop,TS val 218847245 ecr 758433630], leng
th 0
12:23:12.874224 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4182595:4183991, ack 15771, win 379, options [nop,nop,TS val 758433641 ec
r 218847144], length 1396
12:23:12.874259 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4183991, win 4328, options [nop,nop,TS val 218847261 ecr 758433641], leng
th 0
12:23:12.888114 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4183991:4185387, ack 15771, win 379, options [nop,nop,TS val 758433651 ec
r 218847199], length 1396
12:23:12.888151 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4185387, win 4328, options [nop,nop,TS val 218847275 ecr 758433651], leng
th 0
12:23:12.897208 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4185387:4186783, ack 15771, win 379, options [nop,nop,TS val 758433661 ec
r 218847199], length 1396
12:23:12.897234 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4186783, win 4328, options [nop,nop,TS val 218847284 ecr 758433661], leng
th 0
12:23:12.922743 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4186783:4188179, ack 15771, win 379, options [nop,nop,TS val 758433672 ec
r 218847199], length 1396
12:23:12.922770 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4188179, win 4328, options [nop,nop,TS val 218847309 ecr 758433672], leng
th 0
12:23:12.943635 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4188179:4189575, ack 15771, win 379, options [nop,nop,TS val 758433682 ec
r 218847199], length 1396
12:23:12.943669 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4189575, win 4328, options [nop,nop,TS val 218847330 ecr 758433682], leng
th 0
12:23:12.943714 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4189575:4190971, ack 15771, win 379, options [nop,nop,TS val 758433692 ec
r 218847243], length 1396
12:23:12.943737 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4190971, win 4320, options [nop,nop,TS val 218847330 ecr 758433692], leng
th 0
12:23:12.943767 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4190971:4192367, ack 15771, win 379, options [nop,nop,TS val 758433703 ec
r 218847245], length 1396
12:23:12.943778 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4192367, win 4312, options [nop,nop,TS val 218847330 ecr 758433703], leng
th 0
12:23:12.952759 IP 74.125.10.55.https > vulp-nezuko.58254: Flags [.], seq 4192367:4193763, ack 15771, win 379, options [nop,nop,TS val 758433713 ec
r 218847261], length 1396
12:23:12.952788 IP vulp-nezuko.58254 > 74.125.10.55.https: Flags [.], ack 4193763, win 4328, options [nop,nop,TS val 218847339 ecr 758433713], leng
th 0
12:23:13.041697 IP6 fe80::1ad7:17ff:fe66:360d > ip6-allrouters: ICMP6, router solicitation, length 16
```

Рисунок 2.7 – Виведення на консоль даних про поточний процесс захоплення пакетів з TCPDump

Виходячи з того, що утиліта орієнтована на консольне використання, встановлення опцій захоплення пакетів, та виведення тих чи інших відомостей здійснюється з командного рядка. Гнучке конфігурування команд при цьому забезпечується застосуванням параметрів та ключів. На тлі широкого функціоналу утиліти, дана обставина може розглядатися як недолік, оскільки:

- у рамках TCPDump застосовується близько 30 ключів та 7 параметрів;
- фільтри у вигляді заготовок (на кшталт доступних у WireShark) на рівні утиліти не передбачено, замість них використовуються комбінації внутрішніх команд, ключів та параметрів;
- опанування засобом на достатньому рівні потребує деякого часу від адміністратора мережі та/або фахівця з безпеки.

2.5 Загальний базис побудови засобів захоплення пакетів

На сьогоднішній день існує значна кількість рішень для рішення завдання захоплення мережевого трафіку - починаючи з інтегрованої у UNIX-системи Berkeley Packet Filter, закінчуючи Wireshark або SolarWinds Real-Time NetFlow Traffic Analyzer. Утім, усі вони працюють на базі однієї загальної концепції, хоча кожне з рішень використовує свій підхід до реалізації необхідних функцій моніторингу.

З урахуванням сьогоднішніх тенденцій у будь-якій локальній мережі, що має вихід в інтернет, та й не тільки, перегляд мережного трафіку, як вже було зазначено, є важливою а деколи - необхідною умовою для забезпечення розслідування інцидентів, пов'язаних з мережевою безпекою.

Звичайно, використовувати готові рішення для відстеження мережевого трафіку – зручно. Разом з тим, щоб створити сніфер, необхідно розуміти принцип роботи Berkeley Packet Filter, як умовного першоджерела усіх існуючих зараз систем моніторингу.

У свою чергу, Berkeley Packet Filter (BPF) – одна з технолгій ядра UNIX, що призначена для захоплення пакетів; у сутності, являє собою драйвер для передавання та приймання пакетів.

Архітектурно BPF поділяється на два елементи, а саме – мережеву пастку та пакетний фільтр.

Тут мережева пастка – це петлева функція, що є частиною коду BPF, утім, самим BPF вона не викликається. Її виклик виконує драйвер мережевого адаптеру у момент приймання чергового вхідного пакету. Дана функція викликається для кожного пакету, що надходить. Далі, мережева пастка виконує зчитування пакетів, їх копіювання, після чого копії надсилає до додатку верхнього рівня (пакетного фільтру).

При цьому, пакетный фільтр – це реалізований користувачем функціонал, що повертає двійковий результат пакету.

Схему взаємодії BPF з операційною системою може бути представленою таким чином, як це показано рис. 2.8

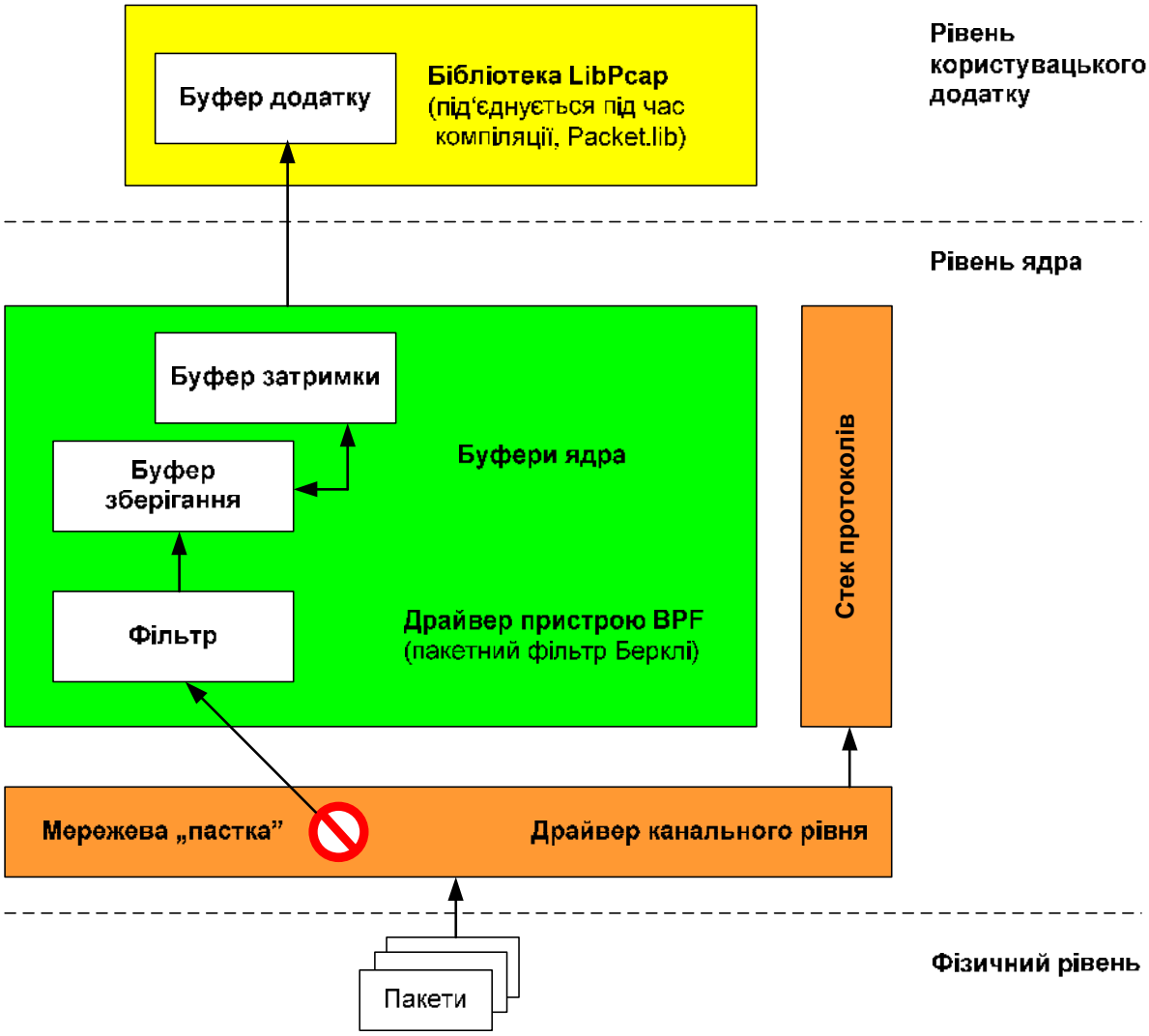


Рисунок 2.8 - Схема взаємодії BPF з операційною системою

Для розробки сніфера доцільно використати API (інтерфейс прикладного програмування) Pcap.

Архітектура Pcap є повністю сумісною з BPF та дає змогу здійснювати фільтрацію пакетів аналогічно BPF, але на рівні користувача. Програмна реалізація Pcap представлена у вигляді NDIS-драйверу, що є специфікацією інтерфейсу мережевого драйверу.

Призначення NDIS – надання можливості драйверу протоколів як приймати, так і передавати пакеты мережею незалежно від операційної системи та місця зберігання.

Разом з тим, NDIS має ряд відмінностей від BPF, а саме - фільтрація здійснюється на рівні користувача, а відтак – кожен вхідний пакет повинен копіюватися з ядра до буферу додатку ще до його фільтрації.

Окрім цього, у ядрі оперційної системи буферизація пакетів є відсутньою. У свою чергу, управління драйвером здійснюється на базі тієї ж бібліотеки libpcap. За рахунок цього забезпечується програмна сумісність двох архітектур.

3. ПІДГОТОВЧІ ОПЕРАЦІЇ, НЕОБХІДНІ ДЛЯ РЕАЛІЗАЦІЇ ПРОЦЕСУ СНІФІНГУ

3.1 Узагальнений сценарій процесу захоплення пакетів

Процес сніфінгу, що реалізується на базі локального вузла, у загальному випадку перебігає за єдиним сценарієм незалежно від середовища застосування. Схематично даний сценарій є таким, як зображено на рис. 3.1.

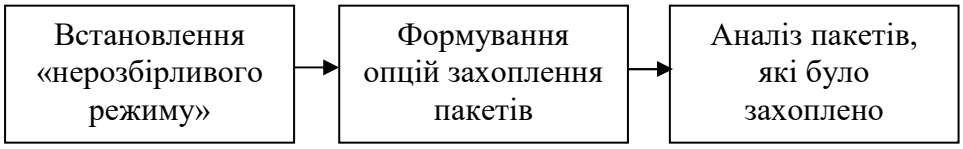


Рисунок 3.1 – Узагальнений сценарій реалізації сніфінгу

Тобто, спочатку мережевий адаптер локального хоста переводиться у нерозбірливий режим, після чого може приймати усі пакети, що транслюються загальним середовищем незалежно від їх адреси призначення.

Далі виконується налаштування механізмів інтерпретації отриманих пакетів, та створення тих чи інших опцій їх захоплення (фільтрів).

У свою чергу, інструментарій управління мережевим адаптером та обробки пакетів надається бібліотекою Pcap (Packet Capture).

3.2 API Pcap та особливості її застосування для побудови сніфера

Pcap являє собою;

- інтерфейс програмування додатків (у даному разі – орієнтований на взаємодію з мережевими адаптерами);
- бібліотеку функцій, призначених для управління мережевим адаптером;
- драйвери NDIS.

На сьогодні Pcap має ряд реалізацій, а саме:

- WinPcap – пакет, розроблений для роботи у Windows-середовищі, де використовуються платформи від Windows 95, до Windows 7;
- Win10Pcap – оновлена версія пакету, що орієнтується на роботу також у Windows-середовищі, з платформами від Windows 7 (останні релізи), до Windows 10;
- LibPcap –пакет засобів для Unix-середовища. з платформами від Windows 7 (останні релізи), до Windows 10.

Порівняльний аналіз можливостей зазначених засобів наводиться табл. 3.1.

Таблиця 3.1 – Порівняльний аналіз особливостей різних реалізацій Pcap

Показник / Засіб	Платформа	Стабільність	Робота з нестандартними атаптерами	Додаткові можливості
WinPcap	Windows 95 - Windows 7	+	-	
Win10Pcap	Windows 7 - Windows 10	Стабільність не гарантується цілково	+	Захоплення тегів IEEE802.1Q
LibPcap	Unix	+	+	

Як видно з табл.3.1, уніфікованого засобу Pcap на сьогодні не існує. Відтак, вибір конкретної версії бібліотеки залежить від середовища її застосування та версії платформи, у рамках якої планується виконувати сніфінг.

У свою чергу, Pcap традиційно орієнтована на використання у середовищі C та C++, так як початково створена на мові C.

Разом з тим, ряд поширених середовищ розробки мають власні локалізації бібліотеки. Так, для мови Python це - Pcapy та Python-libpcap, на випадок Java – NetPcap, JPcap на ін.

У нашому випадку, відповідно до технічного завдання, мова йде про Unix-середовище, а саме - GNU/Linux Ubuntu 22.04. Відтак, використовується LibPcap.

Оскільки дана бібліотека не входить до переліку компонент, що включаються у збірку ОС за замовчуванням, попередньо необхідно виконати її завантаження та надалі – інсталяцію у системі.

3.3 Завантаження та налаштування LibPcap

Бібліотеку LibPcap включено до репозиторію Ubuntu. Таким чином, найбільш доцільний спосіб інтеграції бібліотеки и робоче середовище – скористатися командою:

sudo apt install libpcap-dev

При цьому, одночасно до завантаження libpcap-dev, виконується визначення необхідних залежностей між пакетами автоматично (рис. 3.2).

```
ich@i:~$ sudo apt install libpcap-dev
[sudo] пароль для ich:
Чтение списков пакетов… Готово
Построение дерева зависимостей… Готово
Чтение информации о состоянии… Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
  libflashrom1 libftdi1-2 libllvm13 linux-headers-5.15.0-43 linux-headers-5.15.0-43-generic linux-image-5.15.0-43-generic
  linux-modules-5.15.0-43-generic linux-modules-extra-5.15.0-43-generic
Для их удаления используйте «sudo apt autoremove».
Будут установлены следующие дополнительные пакеты:
  libc-dev-bin libc-devtools libc6-dev libcrypt-dev libdbus-1-dev libdpkg-perl libfile-fcntllock-perl libnsl-dev libpcap0.8-dev
  libtirpc-dev linux-libc-dev manpages-dev pkg-config rpcsvc-proto
Предлагаемые пакеты:
  glibc-doc debian-keyring gcc | c-compiler binutils git bzr dpkg-dev
Следующие НОВЫЕ пакеты будут установлены:
  libc-dev-bin libc-devtools libc6-dev libcrypt-dev libdbus-1-dev libdpkg-perl libfile-fcntllock-perl libnsl-dev libpcap-dev
  libpcap0.8-dev libtirpc-dev linux-libc-dev manpages-dev pkg-config rpcsvc-proto
Обновлено 0 пакетов, установлено 15 новых пакетов, для удаления отмечено 0 пакетов, и 7 пакетов не обновлено.
Необходимо скачать 7 022 kB архивов.
После данной операции объём занятого дискового пространства возрастёт на 31,1 MB.
Хотите продолжить? [Д/н] y
```

Рисунок 3.2 – Процес інсталяції LibPcap

Альтернативним способом інтеграції LibPcap у систему - є завантаженн з сайду розробника www.tcpdump.org архіву libpcap-dev.tar.gz.

У даному випадку, після розпакування архіву, необхідно перейти до отриманої директорії і виконати команду:

make install

На даному кроці процедуру інтеграції бібліотеки LibPcap у систему можна вважати завершеною.

3.4 Початок використання LibPcap

Для побудови програмного коду, який використовує функції, закладені у Libpcap, скористаємося редактором коду Visual Studio Code для Linux Ubuntu.

При цьому, для збірки виконуваного файлу застосовується компілятор gcc.

Спочатку виконаємо ряд необхідних дій, що передують безпосередньому використанню бібліотеки.

Це, зокрема, огляд загальних принципів, на яких базується принцип сніфінгу та базові використовувані функції.

Перш за все, слід розуміти, що процес захоплення пакетів, у сутності, можна сприймати як отримання копій пакетів ще до того, як їх буде оброблено операційною системою.

У свою чергу, LibPcap забезпечує незалежний від реалізації доступ до базового інструментарію перехоплення пакетів, що надається операційною системою.

Сам процес захоплення пакетів реалізується за мірою їх надходження до мережевого адаптеру.

Подальше дослідження можливостей LibPcap виконується в умовах, коли виконується моніторинг трафіку, що надходить з мережі Ethernet.

Спершу, виконаємо опитування активного мережевого адаптеру засобами, які надає LibPcap.

3.4.1 Отримання базової інформації про мережевий адаптер засобами LibPcap

Для побудови відповідного коду необхідно до проекту під’єднати ряд заголовочних файлів, а саме:

- stdio.h;
- stdlib.h;
- pcap.h;
- errno.h;
- sys/socket.h;
- netinet/in.h;
- netinet/if_ether.h

- arpa/inet.h.

За умови, що у системі присунті gcc та LibPcap, означені вище файли є доступними.

Також, попередньо необхідно зарезервувати пам'ять для:

- даних, що надає функція *pcap_lookupdev (char*)*, яка повертає маркер типу char на ім’я використовуваного пристрою. Інакше кажучи, дана функція повертає назву базового мережевого пристрою. У свою чергу, у ролі аргументу функція отримує символьный массив, для того, щоб зберігати у ньому помилки;
- даних, що отримуються від функції *int pcap_lookupnet (const char *, bpf_u_int32 *, bpf_u_int32 *, char *)*,що повертає статус виконання операції запиту. У даному разі до пристрою, ім’я якого задається першим аргументом, виконується запит відносно: IP-адреси (запис виконується до другого аргументу, що передається) та відносно маски мережі (другий передаваний аргумент). При цьому, четвертим аргументом також є змінна помилки;
- даних, які надає функція *char *inet_ntoa (struct in_addr __in)*. Дана функція перетворює адресу, початково представлену у форматі IN до ASCII-вигляду. У даному випадку значення, що повертається, являє собою вказівник на внутрішній масив, що містить відповідний рядок.

Отже, для того, щоб звернутися до мережевого пристрою, який може бути опитано, використаємо функцію *pcap_lookupdev.*

Користуючись зазначеними функціями з бібліотеки LibPcap, спочатку виконаємо просте опитування мережевого адаптеру локального хоста. На даному технологічному етапі передбачається отримання даних стосовно:

- системного імені мережевого адаптеру;
- адреси мережевого пристрою;
- маски.

Додатково для випадку, коли у процесі звернення до мережевого адаптеру виникатиме та чи інша помилка (невідповідність умовам використання функцій – відсутність адаптеру та/або маски/адреси), використається змінна помилки.

Відтак, перший фрагмент коду, який створюється, міститиме, як вже зазначалося, підключення файлів сторонніх бібліотек, а саме:

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <pcap.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
```

Далі, після об’явлення головної частини програми, задаються змінні. Тут змінні для зберігання відомостей, отриманих напряму від мережевого адаптеру, вносимо самостійно. У цьому разі використовується наступний перелік змінних:

- char *dev (передається назва мережевого адаптеру);
- char *net (для зберігання мережевої адреси);
- char *mask (передається маска);
- int ret (код повернення помилки).

Додатково зарезервуємо змінні для функції запиту мережевих параметрів pcap_lookupnet (*bpf_u_int32*) та переводу інформації з видачі до зрозумілого вигляду inet_ntoa.

Таким чином, фрагмент коду, який здійснює пошук, опитування мережевого адаптеру, форматування отриманих даних та виведення інформації у разі виникнення помилки буде наступним:

```
dev = pcap_lookupdev(errbuf);

/* перевірка помилки */
if(dev == NULL)
{
    printf("%s\n", errbuf);
    exit(1);
}

/* виведення назви мережевого адаптеру на консоль */
printf("DEV: %s\n", dev);

/* запит до pcap щодо мережевої адреси та маски пристрою */
ret = pcap_lookupnet(dev, &netp, &maskp, errbuf);

if(ret == -1)
{
    printf("%s\n", errbuf);
    exit(1);
}
```

```
    /* запит мережевої адреси у форматі, зрозумілому для людини
*/
    addr.s_addr = netp;
    net = inet_ntoa(addr);


    if(net == NULL)
    {
        perror("No network connection found");
        exit(1);
    }

    printf("NET: %s\n", net);

    /* запит маски у форматі, зрозумілому для людини */
    addr.s_addr = maskp;
    mask = inet_ntoa(addr);

    if(mask == NULL)
    {
        perror("No network connection found ");
        exit(1);
    }

    printf("MASK: %s\n",mask);

    return 0;
```

Далі виконаємо компіляцію створеного коду.

3.4.2 Компіляція програмного модулю

Для того, щоб забезпечити успішне виконання процедури компіляції, необхідно у терміналі ввести команду вигляду:

gcc <имя_файла>.c –o <имя_файла> -lpcap

Даний процес ілюструється на рис.3.3.

У процесі компіляції слід звернути увагу на те, що:

- обов’язковою умовою тут є внесення у командну конструкцію компілювання, як одного з параметрів, імені бібліотеки *–lpcap*;

- процес компіляції може супроводжуватися попередженнями (warning) системи.

У свою чергу, присутність попереджень від системи жодним чином не впливає на функціональність утвореного модулю та не свідчить про наявність помилок.

Майже завжди попередження виникають тоді. коли у системі є присутніми ряд мережевих інтерфейсів.

Відтак, при цьому пропонується до виконання команда, що дозволить отримати повний перелік мережевих інтерфейсів.

У нашому випадку такі попередження може бути проігноровано, так як функція *pcap_lookupdev* визначає основний мережевий інтерфейс, що є активним, самостійно.

Результатом виконання команди компіляції є генерування файлу в поточній локації.

Ім’я даного файлу буде таким, як було вказано після параметру «–o» командної конструкції компілювання.

У нашому випадку це файл out_net_interface (рис. 3.3).

```
ich@i:~/PCap$ gcc define_network_interface.c -o out_net_interface -
lpcap
define_network_interface.c: In function ‘main’:
define_network_interface.c:21:5: warning: ‘pcap_lookupdev’ is depre
cated: use 'pcap_findalldevs' and use the first device [-Wdeprecate
d-declarations]
   21 |     dev = pcap_lookupdev(errbuf);
      |     ^~~
In file included from /usr/include/pcap.h:43,
                 from define_network_interface.c:3:
/usr/include/pcap/pcap.h:394:18: note: declared here
  394 | PCAP_API char   *pcap_lookupdev(char *)
      |                  ^~~~~~~~~~~~~~
ich@i:~/PCap$ ls -ahl
итого 28K
drwxrwxr-x  2 ich ich 4,0K фев 19 22:50 .
drwxr-x--- 20 ich ich 4,0K фев 19 22:50 ..
-rw-rw-r--  1 ich ich 2,0K фев 19 22:50 define_network_interface.c
-rwxrwxr-x  1 ich ich  16K фев 19 22:50 out_net_interface
```

Рисунок 3.3 - Процедура компіляції та її результат

3.5 Запуск програмного модулю

Оскільки у нашому випадку використовується драйвер, наданий LibPcap, який забезпечує пряму взаємодію з мережевим адаптером, тоді для можливості запуску програмного модулю це слід виконувати з правами суперкористувача, тобто, користувача root.

Отже, попередньо у терміналі необхідно попередньо встановити відповідний привілейований режим, виконавши команду *sudo su,* як показано рисунком 3.4.

Далі запуск програми можливим простим зверенням до скомпільованого файлу командою *./<ім'я_файлу>* (для нашого випадку це ./out_net_interface).

При цьому за умови, що програмний модуль створено та далі – скомпільовано коректно, результат виконання команди буде таким, як зображено на рисунку 3.4.

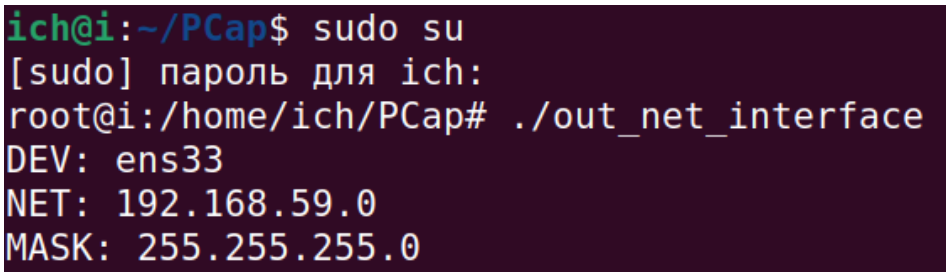


Рисунок 1.4 - Результат виконання програми опитування мережевого адаптеру

3.6 Попередні висновки

У відповідності до чинного технічного завдання, було виконано такі завдання, як:

- огляд існуючої сьогодні концепції сніфінгу пакетів незалежно від середовища;
- аналіз поширених програмних засобів та механізмів, що надають можливість побудови програмних додатків для перехоплення та аналізу трафіку у мережі;
- вибір для подальшого використання бібліотеки LibPcap як такої, що надає достатні можливості для реалізації процесів сніфінгу та дослідження пакетів у середовищі Linux;
- дослідження змісту бібліотеки LibPcap;
- дослідження процесів інтеграції у систему бібліотеки LibPcap на рівні ключового функціоналу;

- побудову простого програмного модулю на базі LibPcap, що дає змогу отримувати відомості щодо мережевого адаптеру та його базисного налаштування (маска та ip-адреса);

- процес компіляції створеного програмного модулю та подальше його виконання.

Далі виконаємо дослідження процесу реалізації механізму захоплення пакетів та їх аналізу.

4. СНІФІНГ ПАКЕТІВ ЗАСОБАМИ LIBPCAP ТА ЇХ НАСТУПНИЙ АНАЛІЗ

4.1 Початок захоплення пакетів

Попередньо засобами, які надає LibPcap, було виконано запит до мережевого адаптеру.

При цьому, технологічний етап опитування мережевого адаптеру розглядався як підготовчий, що передує безпосередньо процесу захоплення пакетів.

Формально, дана операція може бути реалізована додаванням функції *const u_char *pcap_next(pcap_t *, struct pcap_pkthdr *)* з бібліотеки LibPcap.

Утім, додавання зазначеної функції до коду, створеного у п. 3.4.1, не достатньо для реалізації процесу захоплення пакетів трафіку. Це зумовлюється тим, що для початку захоплення пакетів необхідно відкрити пристрій для сніфінгу.

У свою чергу, для ініціалізації початку захоплення пакетів (відкриття мережевого адаптеру), у рамках бібліотеки LibPcap передбачено функцію *pcap_t *pcap_open_live(const char *, int, int, int, char *)*. Як видно з даного запису, даними, що повертає функція, тут є структура *pcap_t*. Ці дані являють собою дескриптор, який повертається програмою.

Разом з тим, першим аргументом функція приймає відомості про мережевий адаптер (dev), який відкривається для захоплення пакетів.

Водночас, другим аргументом тут є *snaplen*, який вказує максимальный розмір (у байтах) пакетів, які будуть підлягати захопленню.

Третій аргумент, а саме – *promisc*, використовується для переведення мережевої карти у т.з. нерозбірливий режим (тобто, коли можливо приймати усі пакети, незалежно від значення поля destination).

Четвертий аргумент функціїі – *to_ms*, який, у свою чергу, лімітує час очікування пакетов до закінчення часу зчитування (у мілісекундах).

Нарешті, останній, п’ятий аргумент, це – *errbuf*, що призначається для зберігання помилки.

З урахуванням цього, сформуємо частину коду, що включає у себе зазначену функцію:

```
int main(int argc, char **argv)
{
    int i;
    char *dev;
    char errbuf[PCAP_ERRBUF_SIZE];
    pcap_t* descr;
    const u_char *packet;
    struct pcap_pkthdr hdr;
    struct ether_header *eptr;

    u_char *ptr; /* для виведення на консоль даних щодо
виявленого обладнання */

    dev = pcap_lookupdev(errbuf);

    if(dev == NULL)
    {
        printf("%s\n",errbuf);
        exit(1);
    }

    printf("DEV: %s\n",dev);

    descr = pcap_open_live(dev,BUFSIZ,0,-1,errbuf);

    if(descr == NULL)
    {
        printf("pcap_open_live(): %s\n",errbuf);
        exit(1);
    }

    /* виконується захоплення пакету з його описом. Для цього
існує конструкція вигляду u_char *pcap_next(pcap_t *p,struct
pcap_pkthdr *h), тому достатньо просто передати дескриптор, який
одержано за результатом виконання запиту pcap_open_live, та
виділену структуру pcap_pkthdr */

    packet = pcap_next(descr,&hdr);

    if(packet == NULL)
    {

        printf("Пакет не було захоплено\n");
```

```
    exit(1);
}
```

При цьому, якщо зазначений код було створено та далі скомпільовано – за результатами виконання програмного модулю у наших умовах отримаємо сповіщення «Пакет не було захоплено».

Даний результат виконання коду зумовлюється тим, що його функціональну частину не завершено.

Відтак, зазначимо, що функція *pcap_next* у ролі другого аргументу отримує структуру pcap_pkthdr.

У свою чергу, pcap_pkthdr включає у себе під-структуру, що містить часову мітку, довжину присутньої частини та довжину самого пакету.

Для того, щоб виокремити зазначені дані, необхідно звернутися до відповдого поля у самій структурі, як це передбачено C++.

У нашому випадку, це може бути реалізовано з використанням конструкцій вигляду:

```
printf("Захоплення пакету довжиною %d\n",hdr.len);
printf("Довжина адреси Ethernet рівна %d\n",ETHER_HDR_LEN);
```

За результатами виконання даних команд ми отримуємо відомості щодо довжини пакету та адреси.

Проте разом з тим, за виконання означених дій все ще не забезпечує отримання відомостей щодо перехопленого пакету.

Відтак, для одержання інформації відносно захопленого пакету необхідно проаналізувати заголовок header, а саме:

```
eptr = (struct ether_header *) packet,
```

тут змінну eptr означено при ініціалізації програмного коду. Дана змінна бере участь у процесі обробки функції *struct ether_header,* результат виконання якої, у свою чергу, містить поля, що надають змогу визначити тип пакету. Який було отримано.

Для цього слугує наступний сегмент коду:

```
/* виконується ряд перевірок для визначення, який саме, тип
пакету  перехоплено */
    if (ntohs (eptr->ether_type) == ETHERTYPE_IP)
    {
        printf("Тип Ethernet hex:%x dec:%d є IP-пакетом\n",
                ntohs(eptr->ether_type),
                ntohs(eptr->ether_type));
    }
    else  if (ntohs (eptr->ether_type) == ETHERTYPE_ARP)
    {
        printf("Тип Ethernet hex:%x dec:%d є пакетом ARP\n",
                ntohs(eptr->ether_type),
                ntohs(eptr->ether_type));
    }
    else
    {
        printf("Тип Ethernet %x не IP", ntohs(eptr-
>ether_type));
        exit(1);
    }
```

У даному разі ключовою функцією є *ntohs*, котра дозволяє виконати перекидання частину заголовку пакета (*ether_type*) для перевірки в API.

За замовчуванням, у випадку співпадання з константами самої API, далі може бути отримано тип пакету (див. попередній сегмент коду).

У розглянутому випадку передбачено можливість визначення пакетів IP та ARP.

У цілому, на даному етапі забезпечується можливість визначення типів деяких пакетів з отриманих (надалі зазначений функціонал може бути розширено включенням відповідних умов до розглянутого фрагменту коду). Далі виконаємо дії, спрямовані на отримання більш розширеної інформації щодо перехоплених пакетів.

У першу чергу – даних, що стосуються:

- адреси призначення пакету;
- адреси джерела пакету.

Слід зазначити, що одержання зазначеної інформації не є проблематичним питанням.

Водночас, додаткових зусиль вимагає інтерпретація одержаної інформації.

У зв’язку з цим, розглянемо наступний сегмент коду:

```
ptr = eptr->ether_dhost;
i = ETHER_ADDR_LEN;
printf(" Адреса призначення:  ");
do{
    printf("%s%x",(i == ETHER_ADDR_LEN) ? " " : ":",*ptr++);
}while(--i>0);
printf("\n");

ptr = eptr->ether_shost;
i = ETHER_ADDR_LEN;
printf(" Адреса джерела:  ");
do{
    printf("%s%x",(i == ETHER_ADDR_LEN) ? " " : ":",*ptr++);
}while(--i>0);
printf("\n");
```

У зазначеному фрагменті коду вказівник *ptr* позиціонується на необхідну область пам’яті у структурі *ether_header*.

У даній структурі інформацію представлено у бінарному вигляді.

Виходячи з цього, позицію вказівника необхідно послідовно змінювати. Для цього використовується цикл do-while. Далі необхідно виконати збірку програми.

4.2 Збірка програми і захоплення пакетів

Після того, як код програми з зазначених раніше фрагментів було зібрано, виконується спроба її тестування, яке, при цьому, не буде успішним, як свідчить рис. 4.1.

Це, у свою чергу, зумовлено тим, що на стадії написання коду один з аргументів функції *pcap_open_live,* а саме - *to_ms,* попередньо встановлено у значення -1:

```
descr = pcap_open_live(dev,BUFSIZ,0,-1,errbuf).
```

У свою чергу, даний аргумент визначає час очікування пакетів у мілісекундах до вичерпання часу зчитування.

```
root@i:/home/ich/PCap# gcc capture_packet.c -o capture_packet -lpcap
capture_packet.c: In function ‘main’:
capture_packet.c:23:5: warning: ‘pcap_lookupdev’ is deprecated: use 'pcap_findal
ldevs' and use the first device [-Wdeprecated-declarations]
   23 |     dev = pcap_lookupdev(errbuf);
      |     ^~~
In file included from /usr/include/pcap.h:43,
                 from capture_packet.c:3:
/usr/include/pcap/pcap.h:394:18: note: declared here
  394 | PCAP_API char   *pcap_lookupdev(char *)
      |                  ^~~~~~~~~~~~~~
root@i:/home/ich/PCap# ./capture_packet
DEV: ens33
Пакет не було захоплено
root@i:/home/ich/PCap#
```

Рисунок 4.1 – Поточне тестування програмного модулю

Таким чином, за вказаний час пакет неможливо зчитати, так як на його зчитування фактично не резервується часу.

Виходячи з цього, доцільно змінити значення аргументру *to_ms*, наприклад, з «-1», на «1000».

Далі, виконавши зазначені дії, та вдруге здійснивши компіляції програми, отримуємо результат, який ілюструється рис. 4.2.

```
root@i:/home/ich/PCap# gcc capture_packet.c -o capture_packet -lpcap
capture_packet.c: In function ‘main’:
capture_packet.c:23:5: warning: ‘pcap_lookupdev’ is deprecated: use 'pcap_findal
ldevs' and use the first device [-Wdeprecated-declarations]
   23 |     dev = pcap_lookupdev(errbuf);
      |     ^~~
In file included from /usr/include/pcap.h:43,
                 from capture_packet.c:3:
/usr/include/pcap/pcap.h:394:18: note: declared here
  394 | PCAP_API char   *pcap_lookupdev(char *)
      |                  ^~~~~~~~~~~~~~
root@i:/home/ich/PCap# ./capture_packet
DEV: ens33
Захоплено пакет довжиною 103
Довжина адреси Ethernet рівна 14
Тип Ethernet hex:800 dec:2048 є IP - пакетом
 Адреса призначення:    0:50:56:fe:ae:d1
 Адреса джерела:       0:c:29:b:b9:ae
root@i:/home/ich/PCap#
```

Рисунок 4.2 - Відображення інформації що міститься у пакеті

Отже, у даному випадку забезпечується функціонування таких механізмів, як:

1. Механізм захоплення пакетів.
2. Механізм аналізу захопленого пакету з визначенням ряду його параметрів, а саме:

- довжина пакету;
- тип пакету;
- адреса джерела пакету та його призначення.

Разом з тим, на поточному етапові створюваний програмний засіб поки що не здатен реалізувати функцію сніфінгу. Річ у тім, що, як було описано у п.4.1, функція *pcap_open_live* на момент розгляду мала аргументи, як показано далі:

```
pcap_open_live(dev,BUFSIZ,0,-1,errbuf)
```

Як бачимо, у даному разі значення параметру prmisc, який знаходиться на третій позиції у розглянутій функції, обрано за замовчуванням, тобто, нульовим.

У цьому випадку нерозбірливий режим роботи мережевого адаптеру не активується – тобто, рис. 4.2 демонструє нам процес перехоплення та наступного аналізу пакетів, що і так було призначено хосту, з якого, власне, виконується моніторинг пакетів.

Відтак, далі змінивши параметр prmisc на будь-яку величину, відмінну від нуля, забезпечується можливість отримувати усі пакети, які може «бачити» мережевий адаптер, незалежно від того, призначаються вони йому, чи ні.

Таким чином, нерозбірливий режим дає змогу перехоплювати усі пакети локальної мережі, якщо поточне налаштування мережевого обладнання дозволяє активувати даний режим (у сучасних мережевих пристроях нерозбірливий режим легко розпізнається і далі блокується).

Разом з тим, оскільки у нашому випадку зазначена проблематика розглядається з позиції адміністратора, чи фахівця з інформаційної безпеки, зазначеними обмеженнями можна знехтувати.

Далі розглянемо реалізацію процесу захоплення пакетів у безперервному режимі.

4.3 Реалізація механізму захоплення довільної кількості пакетів

Очевидно, що будь-який дієвий аналіз мережевого трафіку з використанням програмного засобу, що дозволяє захоплювати один такет у ході одного запуску, неможливий.

Таким чином, для забезпечення рішень усіх пунктів технічного завдання, необхідно реалізувати механізм захоплення довільної кількості пакетів. Наприклад, стільки, скільки буде встановлено користувачем.

У даному випадку знадобиться функція зворотнього виклику, котра передається до *pcap_loop(...)* и далі викликається кожного разу під час отримання пакету.

Загальну логіку даного процесу може бути описано наступним фрагментом коду, де реалізується циклічна функція:

```
    void my_callback(u_char *useless,const struct pcap_pkthdr*
pkthdr,const u_char*packet, int k)
    {
        static int count = 1;
        if (count <=k){
        fprintf(stdout,"%d, ",count);
        fflush(stdout);
        count++;}
        else return;
     }
```

За результатом цього додаток буде захоплювати масив пакетів, не обмежуючись єдиним.

Підсумковий код створеного програмного модулю наводиться у Додатку Б.

ВИСНОВКИ

У відповідності до технічного завдання, під час виконання кваліфікаційної роботи, було здійснено:

- дослідження функціоналу та механізмів, використовуючи які створюються програмні засоби захоплення та дослідження пакетів у мережі;
- розглянуто можливості програмного забезпечення, побудованого на базі Pcap/LibPcap, зокрема – WireShark, IDS Suricata а також утиліта TCPDump та їхня роль у забезпеченні скорочення діючих інформаційних ризиків;
- досліджено можливості та склад пакету LibPcap;
- побудувано консольний додаток, що виконує захоплення та аналіз пакетів у середовищі Linux.

Також, беручи до уваги виключну роль механізмів захоплення та аналізу пакетів у процесі забезпечення комплексної мережевої безпеки, виконано:

- дослідження інформаційних ризиків, які на сьогодні є актуальними;
- дослідження об’єктів та засобів кібератак, зокрема – розвинутих стійких загроз.

При цьому, так як за умовати завдання необхідно було реалізувати програмний засіб, який використовуватиметься у Linux-оточенні, у якості API було застосовано LibPcap.

У свою чергу, програмний засіб реалізує у собі наступний функціонал:

- звернення до мережевого адаптеру та переведення його у нерозбірливий режим;
- зчитування відомостей щодо адрес джерела та призначення пакету;
- розрахунок довжини пакету;
- розпізнавання типової належності перехопленого пакету; даний функціонал далі можу бути розширено внесенням додаткових ознак, що відповідають пакетам інших типів;
- можливість перехоплення та наступного аналізу масиву пакетів, теоретично довільної довжини.

Разом з тим, базові функції, що реалізовано у складі LibPcap, дозволяють також зчитувати зміст перехоплених пакетів у двійковому та шістнадцятеричному форматі представлення.

Таким чином, усі пункти технічного завдання виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Microsoft security report [Електронний ресурс] – Режим доступу: https://microsoft.com/securityinsights.

2. Antivirus and Cybersecurity Statistics & Facts 2021 [Електронний ресурс] – Режим доступу: https://wethegeek.com/antivirus-statistics-facts/

3. Атака drive-download, или тайная загрузка [Електронний ресурс] – Режим доступу: https://book.cyberyozh.com/ru/ataka-drive-download-ili-tajnaya-zagruzka.

4. Microsoft unsecurity solutions [Електронний ресурс] Режим доступу: http://www.seattlepi.com/business/275780_msftsuit29.html

5. Stood, A. Targeted Cyber Attacks. — Elsevier Inc.. — 2014. — ISBN 978-0-12-800604-7.

6. Шаньгін В.Ф. Захист інформації в розподілених корпоративних мережах і системах [Текст]: підручник / В.Ф. Шаньгин, А.В. Соколов. – М.: ДМК Пресс, 2002. – 656 с.

7. Wrightson T. Advanced Persistent Threat Hacking: The Art and Science of Hacking Any Organization. – McGraw-Hill, - 2014. – 464 p. ISBN: 9780071828376

8. Hatfield, Joseph M. Virtuous human hacking: The ethics of social engineering in penetration-testing. Computers & Security. 83: 354–366. doi:10.1016/j.cose.2019.02.012. S2CID 86565713.

9. Donaldson S., Siegel S., Williams C. Enterprise Cybersecurity: How to Build a Successful Cyberdefense Program Against Advanced Threats. – Apress, - 2015. – 578 p.

10. Anderson, Ross J. (2008). Security engineering: a guide to building dependable distributed systems (2 ed.). Indianapolis, IN: Wiley. p. 1040. ISBN 978-0-470-06852-6.

11. Bejtlich R. The Practice of Network Security Monitoring: Understanding Incident Detection and Response / Richard Bejtlich. – San Francisco: Search Press Inc, 2013. – 341 с

12. Олифер, В.Г. Безопасность компьютерных сетей / В.Г. Олифер, Н.А. Олифер. – Москва: Горячая линия-Телеком, 2016. – 643 с.: ил., табл. - ISBN 978-5-9912042-0-0: 38.50.

13. Wireshark - GoDeep [Електронний ресурс] Режим доступу: https://www.wireshark.org/

14. Git-Hub-the-tcpdump-group: the TCPDump network dissector [Електронний ресурс] Режим доступу: https://github.com/the-tcpdump-group/tcpdump/

15. Home - Suricata [Електронний ресурс] Режим доступу: https://suricata.io/

16. GitHub - OISF/suricata: Suricata is a network Intrusion Detection System, Intrusion Prevention System and Network Security Monitoring engine developed by the OISF and the Suricata community [Електронний ресурс] Режим доступу: https://github.com/OISF/suricata

17. Winpcap [Електронний ресурс] Режим доступу: http://www.winpcap.org/

18. Nath A. Packet Analysis with Wireshark. – Packt Publishing, - 2015. – 289 p.

19. Piper S. Network Packet Brokers For Dummies, VSS Monitoring Special Edition. – New Jersey: John Wiley & Sons, Inc, - 2012. – 77 p.

20. Stern K. Lecture Workshop - Understanding the Initiation of Projects within Cyber Insider (CINDER) environments: CINDER Threat Detection using Packet Analysis Taschenbuch – 19. Juni 2007.

21. Libpcap 1.11.0b7 documentation - libpcap 1.11.0b7 documentation [Електронний ресурс] Режим доступу: https://libpcap.readthedocs.io/en/latest/index.html

22. GitHub - the-tcpdump-group/libpcap: the LIBpcap interface to various kernel packet capture mechanism [Електронний ресурс] Режим доступу: https://github.com/the-tcpdump-group/libpcap

23. HOME | TCPDUMP&LIBPCAP [Електронний ресурс] Режим доступу: https://www.tcpdump.org/

24. C++ tutorial [Електронний ресурс] Режим доступу: https://cplusplus.com/doc/tutorial/

25. Learn C++ programming [Електронний ресурс] Режим доступу: https://programiz.pages.dev/cpp-programming/

26. Лапонина О. Р. Межсетевое экранирование. — Бином, 2014. — 343 с. — ISBN 5-94774-603-4.

27. Норткатт, С. Защита сетевого периметра / С.Норткатт. – К.: ООО "ТИД "ДС", 2004. – 672 с.

28. Bock L. Learn Wireshark: A definitive guide to expertly analyzing protocols and troubleshooting networks using Wireshark, 2nd Edition. — Packt Publishing, 2022. — 606 c. — ISBN 5-94774-603-4.

29. Фонтана Л. Калавера Д. BPF для мониторинга Linux. — O'Rielly, 2021. — ISBN: 978-5-4461-1624-9.