ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Метод моніторингу трафіка в корпоративній мережі

кваліфікаційна робота

Виконав:
студент гр. СПм-23-5
Смірнов В.Р.

Керівник:
доц. каф. ЕОМ,
к.т.н. Федорченко В.М.

МЕТА РОБОТИ ТА ЗАВДАННЯ

Метою роботи є розробка, реалізація та апробація методу моніторингу мережного трафіку в корпоративному середовищі, який забезпечує виділення логічних потоків, аналіз протоколів, оцінку ентропії корисного навантаження, виявлення аномалій та побудову візуалізацій для підтримки мережної безпеки.

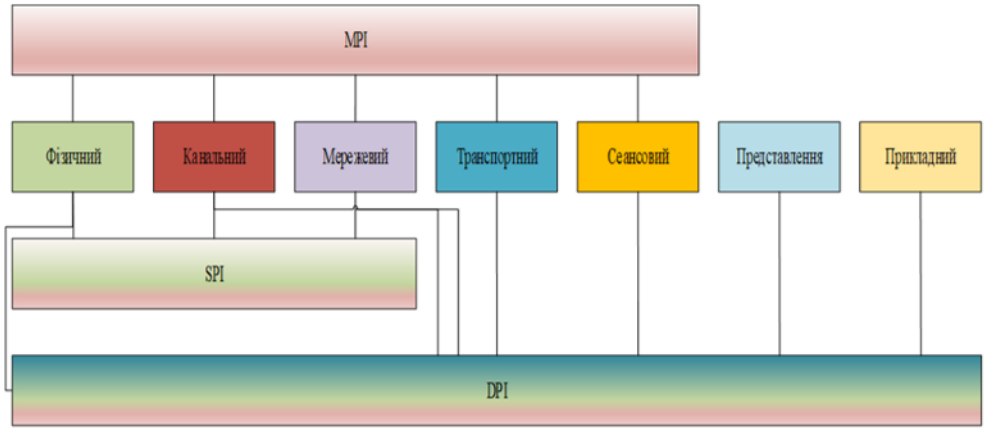
Об'єктом дослідження є процеси передавання, обробки та контролю мережного трафіку в корпоративному середовищі.

Задачі:

- розробити концепцію методу моніторингу, що включає формальні моделі представлення пакетів, логічних потоків, протоколів і їх взаємозв'язків;розробка програмних засобів моніторингу корпоративної мережі з використанням досліджених моделей та методів;
- реалізувати процедуру оцінювання ентропії корисного навантаження для виявлення ймовірного шифрування або ненормативних передач даних;
- інтегрувати механізми виявлення аномалій, базовані на статистичних ознаках потоків: розмір, частота, ентропія, розподілення активності тощо;
- створити прототип програмного засобу в середовищі Google Colab, що включає генерацію тестового трафіку, обробку .pcap-файлів, аналіз логічних потоків та візуалізацію результатів;
- провести експериментальну апробацію розробленого методу на згенерованих даних, включаючи моделювання аномальних сценаріїв, з метою перевірки ефективності розробленого підходу.

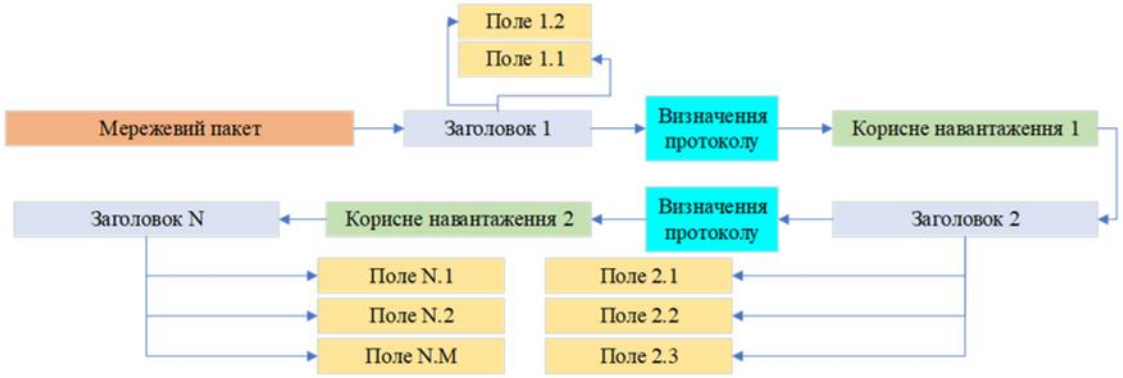
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 3

РІВНІ ДЛЯ ПРОГРАМНИХ ЗАСОБІВ АНАЛІЗУ ТРАФІКА 4



ВИДІЛЕННЯ І РОЗБІР ЗАГОЛОВКІВ ПРОТОКОЛІВ В ПАКЕТІ

5



ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНИХ ЗАСОБІВ МОНІТОРИНГУ МЕРЕЖІ

6

- Зчитування та обробка мережевого трафіку: система повинна підтримувати імпорт трафіку з .pcap-файлів, отриманих з мережевих інтерфейсів або симуляторів, з подальшим розбором заголовків та корисного навантаження.
- Генерація тестового трафіку: забезпечення можливості створення контрольованих сценаріїв мережевої активності (TCP, UDP, ICMP, Raw), з метою тестування алгоритмів виявлення та візуалізації.
- Виділення логічних потоків: автоматичне групування пакетів у логічні з'єднання на основі п'яти ознак (src IP, dst IP, src port, dst port, protocol).
- Побудова структури стеку протоколів: підтримка виявлення вкладених протоколів у рамках кожного потоку та формування відповідного дерева або ієрархії.
- Аналіз корисного навантаження: обчислення статистичних характеристик (розмір, ентропія, частота), необхідних для оцінки типу переданих даних (відкриті чи зашифровані).
- Виявлення аномалій: базовий модуль детекції аномалій на рівні логічних потоків, з урахуванням нетипової тривалості, частоти або ентропії.
- Візуалізація результатів: побудова графів зв'язків між вузлами (IP-адресами), гістограм довжин пакетів, часових графіків активності, ентропійних діаграм та інших візуальних уявлень даних.
- Робота в онлайн та офлайн режимах: підтримка обробки як попередньо зібраного трафіку, так і потокового аналізу в реальному часі при відповідному розширенні.
- Інтерфейс взаємодії з користувачем: інтеграція у середовище Google Colab, забезпечення зручного способу запуску всіх компонентів із одного ноутбука, включаючи автоматизовану генерацію, обробку та виведення результатів.
- Масштабованість та розширюваність: можливість додавання нових протоколів, детекторів аномалій або типів візуалізацій без зміни базової архітектури системи.

ПРОГРАМНІ ЗАСОБИ МОНІТОРИНГУ МЕРЕЖІ ТА АНАЛІЗУ ТРАФІКА 7

Мережна траса	Особливості
Trace11.pcap	перегрупування TCP-сегментів; TCP-з'єднання, встановлені до початку запису траси
Trace12.pcap	перегрупування TCP-сегментів;повторна передача TCP-сегментів
Trace13.pcap	повторна передача TCP-сегментів
Trace14.pcap	повторна передача TCP-сегментів

Мережна траса	Стек протоколів
Trace21.pcap	ETH-IPv4-IPv4-ICMP
Trace22.pcap	ETH-IPv4-GRE-IPv4-ICMP
Trace23.pcap	ETH-IPv4-UDP-Teredo-IPv6-ICMPv6
Trace24.pcap	ETH-VLAN-IPv6-IPv4-GRE-PPP-IPv4-UDP-DNS

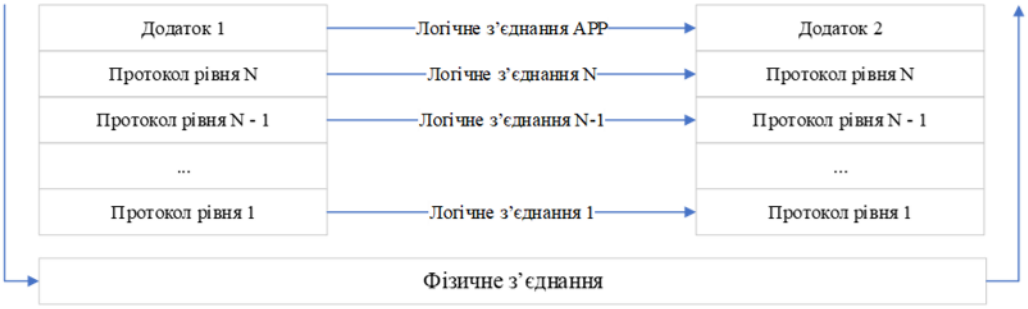
Мережна траса	Особливості
Trace31.pcap	Зашифроване SSL-з'єднання
Trace32.pcap	Передача стиснутих даних за допомогою HTTP

ІСНУЮЧЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 8

Характеристика	Snort	Bro/Zeek	Wireshark
Рік початку розробки	1998	1995	1998
Тип СВВ / інструмента	Сигнатурна СВВ	Гібридна СВВ	Аналізатор трафіку
Метод виявлення / аналізу	Зіставлення з шаблонами	Сигнатури + поведінка + ML	Розбір заголовків і пакетів
Підтримка сигнатур	Так	Так	Ні
Підтримка аномального аналізу	Ні	Так	Ні
Гнучкість налаштувань політик	Обмежена	Висока	Непотрібна
Обробка в реальному часі	Так	Так	Ні
Післязахоплювальний аналіз	Ні	Так	Так
Графічний інтерфейс	Ні	Ні	Так
Кількість підтримуваних протоколів	Обмежено	Широка підтримка	~2000
Можливості фільтрації	Фільтрація на основі правил	Фільтрація подій та політик	~206000 полів для фільтрації
Основне призначення	Виявлення відомих атак	Мережевий моніторинг, інциденти	Детальний аналіз збережених даних

МОДЕЛЬ МЕРЕЖЕВОЇ ВЗАЄМОДІЇ МІЖ ДВОМА ДОДАТКАМИ

9

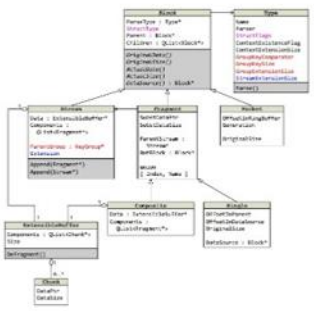


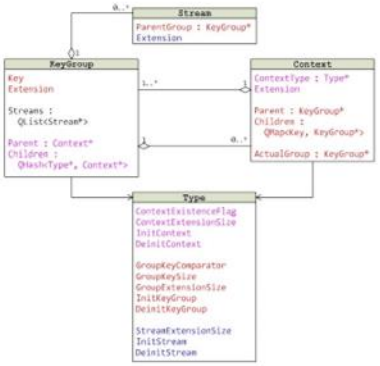
Формальне представлення пакета як кортежу

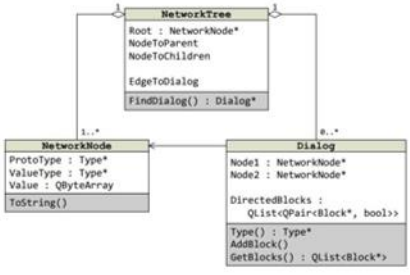
$$Protocol_i\ Packet = (Control_1, Control_2, \ldots, Control_n, Payload)$$

10

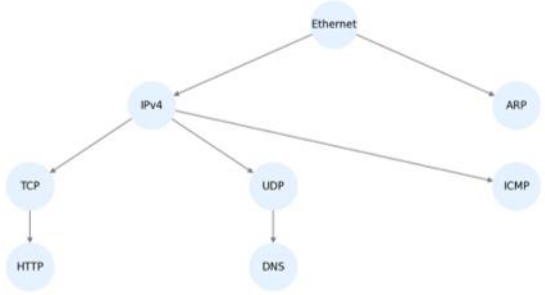
СУТНОСТІ МОДЕЛІ







Приклад дерева контекстів для мережі Ethernet

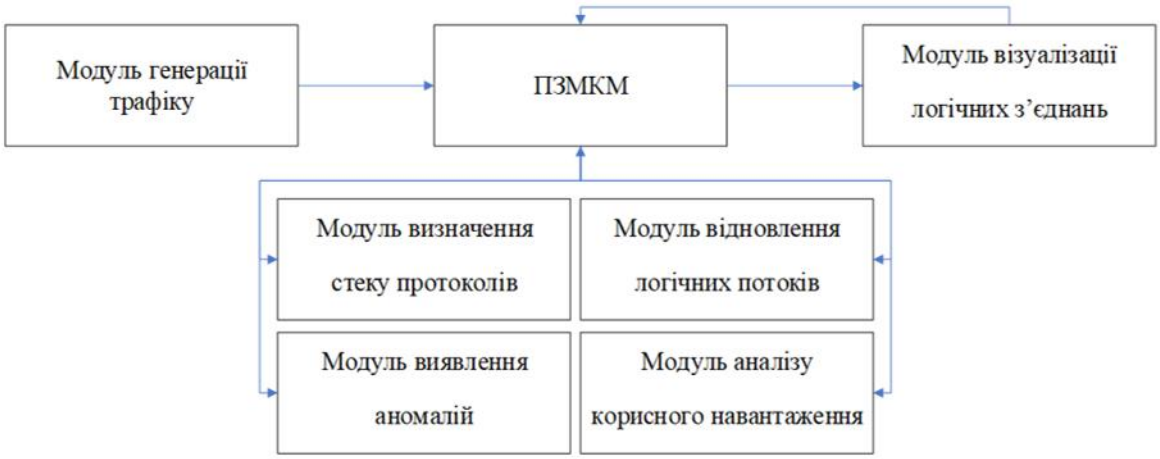


11

МЕТОД МОНІТОРИНГУ ТРАФІКА

12

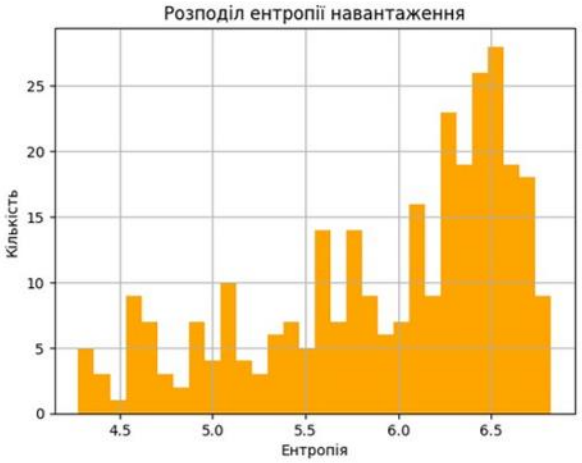
ПРОГРАМНІ ЗАСОБИ МОНІТОРИНГУ КОРПОРАТИВНОЇ МЕРЕЖІ. СХЕМА ВЗАЄМОДІЇ ПРОГРАМНИХ МОДУЛІВ. ВИБІР ПЗ ДЛЯ РОЗРОБКИ



13

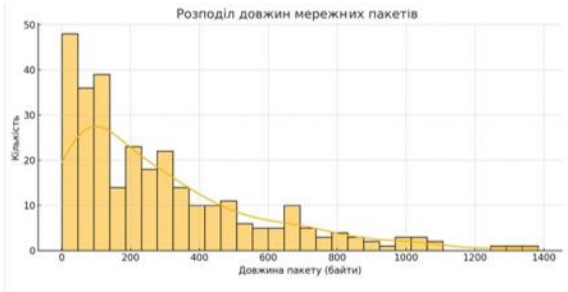
РЕЗУЛЬТАТИ

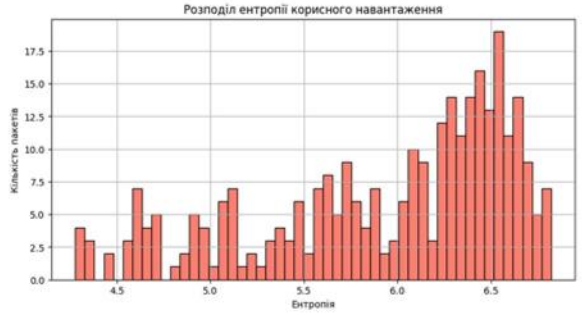
src	dst	sport	dport	proto	packet_count	total_payload_bytes	max_payload_bytes	protocol	avg_payload_bytes
192.168.0.1	192.168.0.2	12345	80	6	5	95	30	TCP	19
192.168.0.1	192.168.0.3	5000	6000	17	8	112	14	UDP	14
192.168.0.2	192.168.0.1	0	0	253	5	100	20	Other	20
192.168.0.2	198.51.100.1	0	0	1	4	32	8	ICMP	8
192.168.0.2	203.0.113.5	4000	4001	17	3	3600	1200	UDP	1200
192.168.0.3	192.168.0.1	0	0	1	30	240	8	ICMP	8
192.168.0.3	198.51.100.1	55555	443	6	10	500	50	TCP	50
192.168.0.4	192.168.0.3	7000	8000	17	5	500	100	UDP	100

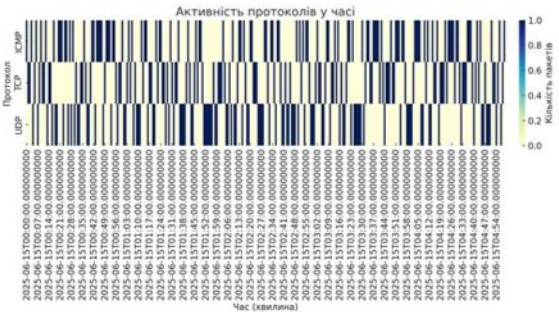


14

РЕЗУЛЬТАТИ







15

АПРОБАЦІЯ

Є. Ю. Глоба[1], В. Р. Смірнов[1], М.С. Нараєвський[1]

[1] Харківський національний технічний університет радіоелектроніки, Харків, Україна

МЕТОД ВИЯВЛЕННЯ АНОМАЛІЙ В КОРПОРАТИВНІЙ МЕРЕЖІ

Анотація. Актуальність. На сучасному етапі розвитку інформаційних технологій та цифрової економіки корпоративні мережі є критично важливими елементами бізнес-інфраструктури. Ефективне функціонування корпоративних мереж забезпечує безперебійну роботу підприємств, високий рівень продуктивності працівників та збереження конфіденційної інформації. Однак із зростанням складності та масштабу мережевих структур зростає й ймовірність загроз, серед яких особливу небезпеку становлять аномальні події, що можуть бути спричинені кібератаками, витоками даних, технічними збоїв або іншими потенційно шкідливими ситуаціями. Актуальність дослідження полягає в тому, що існуючі методи виявлення аномалій нерідко демонструють недостатню точність та оперативність у реальних умовах експлуатації. Тому необхідно розробляти та вдосконалювати методи, які здатні ефективно адаптуватися до мінливих умов сучасних корпоративних мереж і надавати точні результати у режимі реального часу. **Об'єкт дослідження** є мережевий трафік корпоративної інформаційної системи, який аналізується з метою виявлення аномальної поведінки, що потенційно свідчить про кібератаки, порушення політик безпеки або внутрішні загрози. **Мета статті** розробка, реалізація та експериментальне обґрунтування методу виявлення аномалій у корпоративній мережі на основі моделі автоенкодера, який дозволяє ефективно ідентифікувати відхилення від нормального мережевого трафіку без потреби в попередньо мічених даних. **Результати дослідження.** У процесі моделювання було згенеровано дані, які імітують типову та аномальну активність у мережі. Після навчання нейронної мережі на нормальних зразках спостерігалося чітке розділення класів за критерієм помилки реконструкції. Гістограми та boxplot включають на те, що нормальні зразки характеризуються низькими і стабільними значеннями середньоквадратичної помилки, тоді як аномалії – мають виражені відхилення. ROC-крива з AUC = 1.00 підтверджує, що модель здатна безпомилково розрізняти обидві категорії. Матриця плутанини показала практично ідеальне співвідношення між передбаченими та фактичними мітками, що підтверджує високу точність і чутливість моделі. Тренувальна крива свідчить про стабільне навчання без перенавчання, а обраний аномальний поріг дозволив знизити рівень хибнопозитивних спрацювань до мінімуму. Загалом, результати підтверджують, що використання автоенкодера для побудови поведінкової моделі трафіку дозволяє точно та надійно виявляти аномалії в корпоративній мережі, навіть за відсутності мічених прикладів. Запропонований підхід може бути інтегрований у системи моніторингу безпеки для виявлення потенційних активностей у режимі реального часу. **Висновки.** На основі моделі автоенкодера було реалізовано підхід, який дозволяє будувати поведінковий профіль нормального мережевого трафіку та виявляти потенційні відхилення без потреби у мічених даних. Проведені експерименти продемонстрували високу точність класифікації, чітке розділення нормальних і аномальних зразків та позитивну реконструкцію, а також стабільність навчання моделі. Отримані результати підтверджують доцільність використання глибокого навчання для автоматизованого моніторингу мережевої безпеки, а запропонований метод може бути успішно застосований у реальних умовах корпоративної IT-інфраструктури для виявлення інцидентів безпеки в режимі реального часу.

Ключові слова: корпоративна мережа, виявлення аномалій, корпоративна мережа, мережевий трафік, автоенкодер, машинне навчання, кібербезпека, реконструкційна похибка, навчання без вчителя, нейронна мережа, профілювання поведінки, інформаційна безпека.

Вступ

Постановка проблеми. Сучасні корпоративні мережі є невід'ємною складовою цифрової інфраструктури будь-якої компанії, незалежно від її розміру та сфери діяльності. Вони забезпечують ключові процеси, такі як передача інформації, підтримка операційних та управлінських систем, комунікація між підрозділами, а також зовнішню взаємодію з клієнтами й партнерами. В умовах постійного зростання обсягу переданих і оброблюваних даних, а також ускладнення топології мереж, зростають ризики виникнення небезпечних ситуацій, зокрема ...ких, несанкціонованих доступом), так і з різного роду технічними проблемами (відмовами обладнання, помилками програмного забезпечення, неефективністю налаштувань чи конфігурацій). Особливість таких аномалій полягає у тому, що їх важко своєчасно розпізнати за допомогою стандартних інструментів моніторингу й аналізу трафіку, оскільки вони часто не вписуються у попередньо задані моделі та правила, і можуть залишатися прихованими у великих обсягах даних.

Загрози, що виникають у результаті аномальних подій, мають серйозні наслідки, серед яких особливо небезпечними є витоки конфіденційних даних, порушення працездатності бізнес-процесів ... підходи, які використовуються для виявлення таких аномалій. Зокрема, розглядаються методи машинного навчання, глибокого навчання, математичного моделювання.

Також залишається проблема оцінки моделей виявлення аномалій в роботі [3] автора критикують популярні набори даних за низьку репрезентативність і пропонують створення нових сценаріїв на основі реального корпоративного трафіку.

Метою роботи є розробка удосконаленого методу виявлення аномалій у корпоративних мережах, який базується на сучасних підходах у сфері машинного навчання та штучного інтелекту. Запропонований метод має забезпечити високу точність, швидкодію, адаптивність до динамічно змінних умов і низький рівень хибних спрацювань, що є критично важливо для оперативного прийняття рішень та забезпечення надійності функціонування інформаційних систем підприємств

Основний матеріал

Проблема виявлення аномалій у корпоративних мережах уже давно є предметом активного дослідження в галузі інформаційної безпеки. У різних підходах до її вирішення застосовуються як класичні методи аналізу даних, так і сучасні алгоритми машинного навчання та штучного інтелекту [6]. Основна мета таких методів полягає у тому, щоб вчасно виявити незвичну поведінку в мережевому трафіку, яка потенційно може свідчити про загрозу, або про критичні відхилення в роботі системи. У цьому контексті важливою є здатність системи відокремлювати нормальну активність від аномальної, навіть за відсутності явної сигнатури загрози.

Традиційні методи, зокрема сигнатурний аналіз [7], орієнтовані на виявлення вже відомих шаблонів атак, що зберігаються у базі даних сигнатур. Вони ефективні у виявленні відомих загроз, однак позбавлені безсилі перед новими, раніше невідомими формами атак, особливо так званими атаками нульового дня. Крім того, сигнатурні системи вимагають постійного оновлення баз даних і часто створюють великі навантаження на систему безпеки через значну кількість перевірок.

У протилежну сигнатурним методам, аномаліоорієнтовані методи ґрунтуються на побудові моделей «нормальної» поведінки мережі, з якими порівнюється поточна активність. Порушення встановленого шаблону вважається підозрілим і може бути маркером аномалії. Такі методи вимагають етапу навчання або збору статистики, проте вони здатні виявляти

Глоба Є. Ю., Смірнов В. Р., Нараєвський М. С. Метод виявлення аномалій в корпоративній мережі. Системи управління, навігації та зв`язку, вип3. Полтава, 2025. С. 154-158.

16

ВИСНОВКИ

На основі виконаної роботи, яка включала створення, симуляцію, збереження, обробку та візуальний аналіз мережевого трафіку в умовах корпоративної мережі, можна сформулювати низку важливих висновків.

Розроблений метод моніторингу трафіку виявився ефективним для структурованого поділу трафіку на логічні потоки, аналізу вкладених протоколів та обробки корисного навантаження з обрахунком ентропії. Це дозволило не лише ідентифікувати джерела підвищеної мережевої активності, а й оцінити потенційні аномалії без необхідності глибокого інспектування вмісту пакетів, що особливо важливо у випадках використання зашифрованого трафіку.

Аналіз довжин мережевих пакетів засвідчив характерний для корпоративного середовища профіль – переважання коротких службових повідомлень з рідкісними передачами великих пакетів. Це дозволяє зробити висновок про типову інфраструктуру з великою кількістю клієнтських звернень до обмеженої кількості серверів.

17

ВИСНОВКИ

Ентропійний аналіз корисного навантаження вказав на змішаний характер трафіку – одночасну присутність як відкритих, так і потенційно зашифрованих протоколів. Значення ентропії, що коливаються в межах 5.5–6.5 біт, вказують на високу насиченість інформацією, притаманну потокам мультимедіа, VPN або інших зашифрованих сервісів.

Графи логічних з'єднань та розподіл активності за IP-адресами дозволили виокремити як стандартні з'єднання клієнт-сервер, так і потенційно підозрілі концентрації трафіку. Особливо корисною виявилась візуалізація на основі протоколів у часі, яка дозволила простежити інтенсивність і періодичність застосування ICMP та UDP, а також виявити моменти різкого зростання активності – потенційні атаки або автоматизовані скрипти.

ДОДАТОК Б

Програмний код

Б.1 Встановлення бібліотек

```
!pip install scapy pandas matplotlib numpy seaborn networkx
import scapy.all as scapy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
```

Б.2 Генерація симульованого трафіку з різними типами пакетів

```
from scapy.all import Ether, IP, TCP, UDP, ICMP, Raw, wrpcap
packets = []
# Потік 1: TCP нормальний трафік
for i, text in enumerate([
    "Hello, this is normal traffic.",
    "Packet number two of flow1",
    "Just sending some text.",
    "Almost done.",
    "Bye!"
]):
    pkt = Ether() / IP(src="192.168.0.1", dst="192.168.0.2") /
TCP(sport=12345, dport=80, flags="PA") / Raw(text.encode())
    packets.append(pkt)
# Потік 2: TCP з високою ентропією (випадковий payload ~50 байт)
import os
os.environ["PYTHONHASHSEED"] = "0"
np.random.seed(0)  # фіксуємо seed для відтворюваності
for i in range(10):
    random_payload = os.urandom(50)  # 50 випадкових байт
    pkt = Ether() / IP(src="192.168.0.3", dst="198.51.100.1") /
TCP(sport=55555, dport=443, flags="PA") / Raw(random_payload)
    packets.append(pkt)
# Потік 3: UDP нормальний
for i in range(1, 9):
    payload = f"DATA_PACKET_{i:02d}".encode()
    pkt = Ether() / IP(src="192.168.0.1", dst="192.168.0.3") /
UDP(sport=5000, dport=6000) / Raw(payload)
    packets.append(pkt)
# Потік 4: UDP з великим payload (~1200 байт)
pattern = bytes(range(16))  # байти 0x00 ... 0x0F
large_payload = pattern * 75  # 16*75 = 1200 байт
```

```
for i in range(3):
    pkt = Ether() / IP(src="192.168.0.2", dst="203.0.113.5") /
UDP(sport=4000, dport=4001) / Raw(large_payload)
    packets.append(pkt)
# Потік 5: ICMP нормальний ping (4 пакети з різними 8-байтовими
даними)
ping_payloads = [b'abcdefgh', b'ijklmnop', b'qrstuvwx',
b'yzABCDEF']  # різні рядки по 8 байт
for payload in ping_payloads:
    pkt = Ether() / IP(src="192.168.0.2", dst="198.51.100.1") /
ICMP(type=8, code=0) / Raw(payload)
    packets.append(pkt)
# Потік 6: ICMP flood (30 пакетів ping)
# Згенеруємо payload, що циклічно перебирає букви і цифри, для
невеликої варіації
alphabet =
(b"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345678
9")
payload_stream = alphabet * 5  # 62*5 = 310 байт
for i in range(30):
    # беремо 8 байт з потоку (після 248 байт цикл алфавіту
повториться)
    payload = payload_stream[i*8 : i*8+8]
    pkt = Ether() / IP(src="192.168.0.3", dst="192.168.0.1") /
ICMP(type=8, code=0) / Raw(payload)
    packets.append(pkt)
# Потік 7: Raw IP з невідомим протоколом (proto=253)
for i in range(5):
    raw_payload = os.urandom(20)
    # Задаємо proto=253 вручну в IP, додаючи Raw-повідомлення
    pkt = Ether() / IP(src="192.168.0.2", dst="192.168.0.1",
proto=253) / Raw(raw_payload)
    packets.append(pkt)
# Потік 8: UDP з низькою ентропією (payload з однакових байтів
'A')
low_payload = b"A" * 100
for i in range(5):
    pkt = Ether() / IP(src="192.168.0.4", dst="192.168.0.3") /
UDP(sport=7000, dport=8000) / Raw(low_payload)
    packets.append(pkt)
print(f"Загальна кількість згенерованих пакетів:
{len(packets)}")
```

Б.3 Збереження трафіку в .pcap файл. Зчитування пакету з .pcap, обробка та агрегація в CSV

```
# Збереження згенерованих пакетів у PCAP-файл
wrpcap("simulated_traffic.pcap", packets)
print("Файл simulated_traffic.pcap збережено.")
```

Б.4 Зчитування пакету з .pcap, обробка та агрегація в CSV

```
from scapy.all import rdpcap
# Зчитуємо пакети з pcap-файлу
packets_loaded = rdpcap("simulated_traffic.pcap")
# Ініціалізуємо список для зведених даних по потоках
flows = {}
for pkt in packets_loaded:
    # Витягуємо рівень IP
    if not pkt.haslayer(IP):
        continue  # пропустити, якщо відсутній IP-рівень
(неочікувано у нашому випадку)
    ip = pkt[IP]
    src = ip.src
    dst = ip.dst
    proto = ip.proto  # числовий код протоколу (6=TCP, 17=UDP,
1=ICMP, інші)
    # Визначимо порти, якщо це TCP/UDP, інакше 0
    sport = pkt.sport if pkt.haslayer(TCP) or pkt.haslayer(UDP)
else 0
    dport = pkt.dport if pkt.haslayer(TCP) or pkt.haslayer(UDP)
else 0
    # Формуємо ключ потоку
    flow_key = (src, dst, sport, dport, proto)
    # Ініціалізуємо запис для нового потоку
    if flow_key not in flows:
        flows[flow_key] = {
            "src": src, "dst": dst,
            "sport": sport, "dport": dport, "proto": proto,
            "packet_count": 0,
            "total_payload_bytes": 0,
            "max_payload_bytes": 0
        }
    # Оновлюємо метрики потоку
    flows[flow_key]["packet_count"] += 1
    # Обчислюємо довжину корисного навантаження пакета
    payload_len = len(pkt[Raw].load) if pkt.haslayer(Raw) else 0
    flows[flow_key]["total_payload_bytes"] += payload_len
    if payload_len > flows[flow_key]["max_payload_bytes"]:
        flows[flow_key]["max_payload_bytes"] = payload_len
# Перетворюємо flows словник у DataFrame
flow_df = pd.DataFrame(list(flows.values()))
# Додаємо зручну назву протоколу
proto_map = {6: "TCP", 17: "UDP", 1: "ICMP"}
flow_df["protocol"] =
flow_df["proto"].map(proto_map).fillna("Other")
# Розраховуємо середній розмір payload на пакет
flow_df["avg_payload_bytes"] = flow_df["total_payload_bytes"] /
flow_df["packet_count"]
# Зберігаємо у CSV
flow_df.to_csv("flows_summary.csv", index=False)
flow_df.head(10)
```

Б.5 Метод моніторингу та виявлення аномалій на рівні потоків

```
import math

# Функція для обчислення ентропії байтової послідовності (0-1
нормалізована)
def shannon_entropy(data_bytes):
    if len(data_bytes) == 0:
        return 0.0
    # рахуємо частоти кожного байта
    counts = np.bincount(np.frombuffer(data_bytes,
dtype=np.uint8), minlength=256)
    probs = counts[counts > 0] / len(data_bytes)
    entropy = -np.sum(probs * np.log2(probs))
    return entropy / 8  # поділити на 8 (макс ентропія на байт)

# Додаємо колонки з ентропією, унікальними байтами та %
друкованих символів
flow_df["entropy"] = 0.0
flow_df["printable_ratio"] = 0.0

for i, row in flow_df.iterrows():
    # Збираємо всі payload-байти потоку
    # Пакети цього потоку можна відфільтрувати з packets_loaded
    src, dst, sport, dport, proto = row["src"], row["dst"],
row["sport"], row["dport"], row["proto"]
    packets_in_flow = [pkt for pkt in packets_loaded
                       if pkt[IP].src == src and pkt[IP].dst ==
dst
                       and (pkt.sport if pkt.haslayer(TCP) or
pkt.haslayer(UDP) else 0) == sport
                       and (pkt.dport if pkt.haslayer(TCP) or
pkt.haslayer(UDP) else 0) == dport
                       and pkt[IP].proto == proto]
    payload_bytes = b"".join([pkt[Raw].load if pkt.haslayer(Raw)
else b"" for pkt in packets_in_flow])
    flow_df.at[i, "entropy"] = shannon_entropy(payload_bytes)
    # Частка символів 0x20-0x7E (друковані ASCII)
    if len(payload_bytes) > 0:
        printable_count = sum(32 <= b < 127 for b in
payload_bytes)
        flow_df.at[i, "printable_ratio"] = printable_count /
len(payload_bytes)
    else:
        flow_df.at[i, "printable_ratio"] = 0.0

# Встановимо пороги для аномалій
HIGH_PKT_COUNT = 20
LARGE_PAYLOAD = 1000
HIGH_ENTROPY = 0.9
LOW_ENTROPY = 0.2
```

```
# Визначимо підозрілі потоки
flow_df["anomaly_reasons"] = ""
for i, row in flow_df.iterrows():
    reasons = []
    if row["proto"] not in [1, 6, 17]:  # не ICMP, не TCP, не
UDP
        reasons.append("Unknown Protocol")
    if row["packet_count"] > HIGH_PKT_COUNT:
        reasons.append("High Packet Count")
    if row["max_payload_bytes"] > LARGE_PAYLOAD:
        reasons.append("Large Payload Size")
    if row["entropy"] > HIGH_ENTROPY:
        reasons.append("High Payload Entropy")
    if row["entropy"] < LOW_ENTROPY and
row["total_payload_bytes"] > 0:
        reasons.append("Low Payload Entropy")
    # Об'єднуємо причини в рядок
    flow_df.at[i, "anomaly_reasons"] = ", ".join(reasons)

# Виведемо підозрілі потоки
suspicious_flows = flow_df[flow_df["anomaly_reasons"] != ""]
print("Підозрілі потоки:")
for i, flow in suspicious_flows.iterrows():
    proto_name = proto_map.get(flow["proto"],
f"Proto{int(flow['proto'])}")
    if proto_name == "ICMP" or (flow["sport"] == 0 and
flow["dport"] == 0):
        flow_id = f"{flow['src']} -> {flow['dst']}
({proto_name})"
    else:
        flow_id = f"{flow['src']}:{int(flow['sport'])} ->
{flow['dst']}:{int(flow['dport'])} ({proto_name})"
    print(f"- {flow_id} – Reasons: {flow['anomaly_reasons']}")
```

Б.6 Візуалізація результатів моніторингу

```
# Підрахунок пакетів за протоколами
protocol_counts = flow_df["protocol"].value_counts()  # рахуємо
потоки; нам потрібно за пакетами
# Краще порахувати напряму з packets_loaded
proto_names = []
for pkt in packets_loaded:
    proto = pkt[IP].proto
    proto_names.append(proto_map.get(proto, "Other"))
proto_series = pd.Series(proto_names, name="Protocol")
protocol_counts = proto_series.value_counts()
# Будуємо стовпчиковий графік
plt.figure(figsize=(6,4))
sns.barplot(x=protocol_counts.index, y=protocol_counts.values,
palette="tab10")
plt.xlabel("Protocol")
```

```
plt.ylabel("Number of Packets")
plt.title("Packet Count by Protocol Type")
plt.show()
# Обчислюємо ентропію для кожного пакета
packet_entropies = []
for pkt in packets_loaded:
    data = pkt[Raw].load if pkt.haslayer(Raw) else b""
    packet_entropies.append(shannon_entropy(data))
# Гістограма ентропії пакетів
plt.figure(figsize=(6,4))
sns.histplot(packet_entropies, bins=10, color='purple')
plt.xlabel("Entropy (per byte)")
plt.ylabel("Number of Packets")
plt.title("Distribution of Packet Payload Entropy")
plt.show()
# Створюємо граф та додаємо вузли і ребра
G = nx.DiGraph()
color_map = {"TCP": "blue", "UDP": "green", "ICMP": "orange",
"Other": "red"}
for _, row in flow_df.iterrows():
    proto_label = proto_map.get(row["proto"], "Other")
    G.add_edge(row["src"], row["dst"], protocol=proto_label,
color=color_map[proto_label])

# Розрахуємо розташування вузлів для красивого відображення
pos = nx.spring_layout(G, seed=42)
plt.figure(figsize=(8,6))
# Наносимо вузли
nx.draw_networkx_nodes(G, pos, node_size=800,
node_color='lightgray')
# Підписи вузлів (IP-адреси)
nx.draw_networkx_labels(G, pos, font_size=9)
# Наносимо ребра з кольорами та стрілками
edges = G.edges(data=True)
edge_colors = [attr['color'] for _, _, attr in edges]
nx.draw_networkx_edges(G, pos, arrows=True,
edge_color=edge_colors, arrowstyle='-|>', arrowsize=15)
plt.title("Network Flows Graph (edges colored by protocol
type)")
plt.axis('off')
plt.show()
plt.figure(figsize=(6,5))
for _, row in flow_df.iterrows():
    x = row["packet_count"]
    y = row["entropy"]
    size = row["total_payload_bytes"] / 10  # масштабування
розміру
    if row["anomaly_reasons"]:
        plt.scatter(x, y, s=size, c='red', marker='X',
label='Anomalous flow' if 'Anomalous flow' not in
plt.gca().get_legend_handles_labels()[1] else "")
    else:
        plt.scatter(x, y, s=size, c='blue', marker='o',
```

```
label='Normal flow' if 'Normal flow' not in
plt.gca().get_legend_handles_labels()[1] else "")
plt.xlabel("Packet Count")
plt.ylabel("Payload Entropy")
plt.title("Flows by Frequency and Entropy (point size ∝ data
volume)")
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
plt.show()
```