

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

" _____ " _____ 2023 р.

Зав. кафедри, к.т.н., доцент

_____ Лішук Р.І.

КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА

МОДЕЛЮВАННЯ МЕРЕЖЕВОГО ТРАФІКУ ДЛЯ
ЗДІЙСНЕННЯ МОНІТОРИНГУ ТА АНАЛІЗУ
КОМП'ЮТЕРНОЇ МЕРЕЖІ

Виконав – студент 2 курсу, групи 21к-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки

Заєць Юрій Олександрович

Науковий керівник – к.е.н., доцент

Концеба Сергій Михайлович

Умань – 2023

ЗМІСТ

ВСТУП	4
РОЗДІЛ 1. АНАЛІЗ ТА МОНІТОРИНГ МЕРЕЖЕВОГО ТРАФІКУ	6
1.1 Аналіз мережевого трафіку: управління через АРМ	6
1.2 «Глибина» аналізу мережевого трафіку	10
1.3 Особливості передачі мережевого трафіку	14
РОЗДІЛ 2. ЗАСОБИ МОНІТОРИНГУ МЕРЕЖЕВОГО ТРАФІКУ	19
2.1 Пакетні сніфери	19
2.2 Огляд інструментів обробки пакетів	22
2.3 Порівняльний аналіз інструментів обробки пакетів	28
2.4 FATT(Fingerprint all the things)	34
РОЗДІЛ 3. МОДЕЛЮВАННЯ МЕРЕЖЕВОГО ТРАФІКУ	39
3.1 Огляд генераторів мережевого трафіку	39
3.2 Формування ідеї	40
3.3 Прототип генератора мережевого трафіку	42
3.4 Методи оцінки генератора трафіку	49
3.5 Оцінка генератора трафіку	51
ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58
ДОДАТКИ	61

ВСТУП

Активний розвиток інформаційних технологій робить їх невід'ємною частиною сьогодення. Інформаційні системи сьогодні широко експлуатуються як і в комерційних, так і в державних. Їх взаємодія між собою відбувається через мережі.

Спостерігається стрімкий зріст об'єму мережевого трафіку, що провокує ускладнення його структури. Аналіз трафіку стає все більш потрібним в областях контролю та управління, оптимізації, захисту від шкідливого втручання. Моніторинг трафіку також необхідний, щоб більш ефективно діагностувати і вирішувати проблеми, коли вони постають, таким чином не доводячи різні мережеві сервіси до простою впродовж тривалого часу. Доступно багато різних інструментів, які дозволяють допомогти адміністраторам з моніторингом та аналізом мережевого трафіку. Дана робота висвітлює методи моніторингу мережевого трафіку, надає порівняльну характеристику засобів мережевого трафіку, та досліджує методи моделювання мережевого трафіку.

Моніторинг мережі - складне завдання, що вимагає великих сил та витрат, яке є невід'ємною складовою повсякденного життя мережевих адміністраторів. Адміністратори постійно прагнуть підтримати безперебійну роботу своєї мережі. Якщо мережа «впаде» хоча б на невеликий період часу, продуктивність в компанії скоротиться і (в разі організацій які надають державні послуги) сама можливість надання основних послуг буде поставлена під загрозу. У зв'язку з цим адміністраторам необхідно проводити постійний аналіз мережевого трафіку, стежити за його рухом і продуктивністю на всій мережі і перевіряти, чи з'явилися в ній проломи в безпеці. В випадках моніторингу при виявленні проблеми після її вирішення постає питання тестування коректності та цілісної роботи системи для цього в допомогу приходить моделювання трафіка, що дозволяє змодельовати трафік для здійснення аналізу, щоб зробити висновки можливих вразливостей системи, її продуктивності, дослідженні різних

протоколів, алгоритмів або топографій мереж. Це обумовлює актуальність обраної теми.

Метою даної роботи є проведення порівняльного аналізу існуючих засобів моніторингу мережевого трафіку, створення власного аналізатора з базовим функціоналом, дослідження методів моделювання мережевого трафіку.

Результатами даної роботи є власний аналізатор трафіку з базовим функціоналом на основі порівняльного аналізу та алгоритм моделювання мережевого трафіку зі збереженням послідовності.

РОЗДІЛ 1. АНАЛІЗ ТА МОНІТОРИНГ МЕРЕЖЕВОГО ТРАФІКУ

1.1. Аналіз мережевого трафіку: управління через АРМ

Аналіз мережевого трафіку набуває все більшу актуальність у зв'язку з розвитком мережевих технологій, збільшенням обсягу даних, переданих по мережі, впровадженням великої кількості нових мережевих протоколів (в тому числі закритих). В якості основних областей практичного застосування можна виділити наступні:

- виявлення проблем в роботі мережі;
- тестування(налагодження мережевих протоколів);
- запобігання мережевих атак;
- класифікація трафіку.

Поглиблений аналіз трафіку в даний час є однією з основних технологій при вирішенні практичних завдань пов'язаних із забезпеченням безпеки мереж, оптимізацією пропускної здатності каналів передачі даних, контролем якості зв'язку та інших. Якісне рішення згаданих завдань вимагає максимальної глибини розбору мережевого трафіку і надання результатів розбору в зручній формі[12].

Можливість аналізу трафіку на рівні контенту додатків є у вузького кола закритих комерційних систем таких фірм, як Checkpoint і Cisco. Ці системи не дозволяють користувачам вільно розширювати перелік підтримуваних мережевих протоколів, їх закритість ускладнює інтеграцію. У той же час існуючі вільно поширювані інструменти не забезпечують належної якості проведеного аналізу, внаслідок чого з їх допомогою не завжди вдається отримати дані протоколів прикладного рівня.

Аналіз трафіку — необхідна область для управління, контролю і оптимізації, а також для захисту систем. Сучасні корпоративні мережі стають все більш складними завдяки додаванню численних персональних пристроїв, безлічі хмарних додатків, точок доступу, віртуальних серверів тощо. Команди

адміністраторів повинні ретельно керувати пропускнуою здатністю та якістю обслуговування, впроваджувати балансири навантажень для високої доступності, створювати надмірне середовище для резервного копіювання та забезпечувати синхронізовану роботу декількох компонентів у гібридному середовищі.

Аналіз можна здійснювати на рівні окремого пакету, в контексті роботи додатку або в діагностиці складних систем.

Для якісного управління додатком розберемо, що собою являє APM(Application Performance Management).

APM — це процес, який забезпечує зв'язок між роботою програми та всієї структури, в яку додаток входить. Також виміряє, оцінює і документує основні зв'язки додатку. Тобто в додатку ми повинні не лише підраховувати час роботи або використаний трафік, а у разі помилок системи повинні знайти їх причину та знайти методи їх усунення.

Також необхідно враховувати, що всі процеси повинні відбуватись одночасно для мережевого обладнання, каналів зв'язку, серверів тощо.

На базі процесу APM можна виділити види моніторингу:

1. На основі метрик додатків - кілька інструментів використовують різні показники сервера та додатків і називають це APM. У кращому випадку вони можуть повідомити, скільки запитів отримує ваша програма та потенційно, які URL-адреси можуть бути повільними. Оскільки вони не займаються профілюванням рівня коду, вони не можуть сказати вам, чому.

2. Ефективність на рівні коду - Stackify Retrace, New Relic, AppDynamics і Dynatrace - типовий тип APM, на основі кодування профілю та відстеження транзакцій.

3. На основі мережі - ExtraHop використовує термін APM стосовно їх здатності вимірювати продуктивність додатків на основі мережевого трафіку. Існує ціла категорія товарів під назвою NPM, яка зосереджена на цьому типі рішень.

Ось деякі ключових особливостей, яких притримуються більшість розробників при обробці даних:

1. Виконання кожного веб-запиту та транзакції - в основі АРМ ви маєте змогу оцінювати ефективність кожного веб-запиту та транзакції у вашій програмі. Потім ви можете скористатися цим, щоб зрозуміти, до яких запитів програма звертається найбільше, які найповільніші та які саме слід додати для покращення роботи.

2. Профілювання продуктивності на рівні коду - якщо ви хочете зрозуміти, чому ваша програма повільна, викидає помилки або є в ній дивні помилки, вам доведеться перейти до рівня коду. Знаючи, що певний веб-запит не працює, важливо з'ясувати, чому, іноді це дуже складно.

3. Використання та ефективність усіх залежностей програми, таких як бази даних, веб-сервіси, кешування тощо - чому ваш додаток повільний, зазвичай зводиться до швидкого трафіку або до проблеми з однією із ваших програм. Дуже часто зустрічаються такі проблеми:

- Конкретний запит SQL відбувається повільно
- Сервер баз даних SQL не працює
- Сусіди в хмарі викликають проблеми

4. Деталізація окремих веб-запитів чи транзакцій - вирішувати проблеми на виробництві дуже важко. Сліди транзакцій значно полегшують цю роботу, оскільки ви можете бачити деталі про те, що саме відбувається у вашому коді та як це впливає на ваших користувачів.

5. Основний моніторинг сервера та показники, такі як процесор, пам'ять тощо - проблеми із додатком можуть виникати з багатьох причин. Завдяки віртуалізації та хмарі сервер, який опускається, не є настільки поширеним в ці дні. Однак це все ж таки відбувається і це те, за чим потрібно стежити. Також важливо стежити за такими речами, як CPU сервера та пам'ять. Багато сучасних веб-додатків зазвичай не пов'язані з CPU, але вони все ще можуть використовувати багато CPU, і це корисний показник для автоматичного масштабування вашої програми в хмарі.

6. Показники рамки програми, такі як лічильники продуктивності, JMX MBeans тощо - такі показники сервера, як CPU та пам'ять, цікаві, але для розробників показники додатків можуть бути набагато більш цінними для моніторингу справжньої роботи програми. Розробникам необхідно відстежувати показники навколо таких предметів, як збирання сміття, чергування запитів, обсягів транзакцій, часу завантаження сторінки та багато іншого. Також вони можуть стежити за різноманітними лічильниками продуктивності Windows та JMX MBeans. Це також може бути критично важливим для моніторингу таких речей, як Redis, Elasticsearch, SQL та інших сервісів за ключовими показниками.

7. Показники спеціальних програм, створені командою розробників або бізнесом - стандартні показники сервера та додатків можуть бути дуже корисними для моніторингу ваших програм. Однак ви можете отримати набагато більшу цінність, створивши та відстежуючи власні спеціальні показники.

8. Дані журналу додатків-дані журналу - це важлива частина роботи розробників для відслідковування запитів, коли їх програми розгорнуті. Розробникам потрібен доступ до своїх журналів через централізоване рішення для ведення журналів, як продукт управління журналом. На щастя, управління журналом - це включена функція APM в Retrace.

9. Помилки програми-як розробники, нам потрібно знати, коли у користувача відбувається помилка на сторінці, і важливо також знати, яка саме. Помилки - це перша лінія захисту для пошуку проблем із застосуванням [4]. Нам потрібно знайти та виправити помилки або принаймні знати про них, перш ніж клієнти повідомлять нам про це самі, або перестануть бути нашими користувачами.

Відмінне відстеження помилок, звітування та оповіщення є надзвичайно важливими для розробників в системі управління продуктивністю додатків. Я настійно рекомендую налаштувати сповіщення про нові винятки, а також для

моніторингу загальної кількості помилок. Швидше за все, ви знайдете якісь нові помилки, які зможете швидко визначити та виправити.

10. Реальний моніторинг користувачів (RUM) - зрозуміння продуктивності ваших програм на стороні сервера важливо. Однак сьогоденні програми використовують стільки javascript, що важливо також відстежувати, скільки часу їх браузеру потрібно повністю завантажувати та відтворювати ваші веб-сторінки. Проста помилка JavaScript або повільне завантаження файлу javascript може повністю зіпсувати вашу програму. Справжній моніторинг користувачів або RUM - ще одна важлива особливість APM, що розробникам потрібно повністю контролювати свої програми.

1.2.«Глибина» аналізу мережевого трафіку

Розглядаються мережі з комутацією пакетів. Рішення практичних завдань аналізу полягає в розборі заголовків мережевих протоколів в пакетах і відновленні потоків даних, що передаються. Можна виділити 3 групи при проведенні аналізу трафіку(рисунок 1.1): - поверхневий (SPI — Shallow Packet Inspection); - помірний (MPI — Medium Packet Inspection); - поглиблений (DPI — Deep Packet Inspection).

Технологія аналізу трафіку, яка ґрунтується виключно на заголовках пакету рівнів L1-L3 по моделі OSI. Потребує низькі вимоги до обчислювальних ресурсів, що дозволяє аналізувати великі обсяги трафіку. Технологія широко поширена, на її основі працює більшість міжмережевих екранів операційних систем (зокрема в ОС Windows XP / Vista і OS X), маршрутизаторів і інших мережевих пристроїв.[11] На її основі реалізовані мережеві списки контролю доступу на рівні IP адрес і портів (Access Control List, ACL). Таким чином, дана технологія добре підходить для розмежування доступу ззовні до окремих комп'ютерів (IP) і сервісів (порти) внутрішньої мережі.



Рис. 1.1. Відповідність рівнів моделі «OSI» «глибині» аналізу мережевого трафіку

Технологія аналізу трафіку, яка ґрунтується на інспектуванні сесій і сеансів зв'язку, ініційованих додатком, але встановлюваних шлюзом-посередником (Рис. 1.2.). Також застосовується термін «проксі додатків» (application proxy). В рамках даної технології вміст пакетів аналізується частково і по визначеним правилам. Не використовуються складні методи аналізу типу сигнатурного. Пристрої, що реалізують даний функціонал розміщуються між провайдером інтернету і кінцевим користувачем. Дані пристрої розбирають заголовки аж до транспортного рівня і невелику частину даних пакета для зіставлення розібраної частини з деяким списком розбору (parse list), з подальшою реакцією в разі їх виявлення. Дані списки зазвичай коротше списків ACL і надають більш широкий діапазон дій на відміну від «дозволити / заборонити» в разі ACL. Ці списки також більш виразні, так як дозволяють прив'язуватися не до IP-адреса, а до формату даних пакетів і даними деяких протоколів рівня додатки, наприклад, URL-адресів в разі протоколу HTTP. За допомогою MPI можна, наприклад, заблокувати

можливість отримання flash-файлів або картинок з певних інтернет сервісів (на рівні уявлення OSI) або заблокувати частину команд (на рівні додатку OSI) в окремих протоколах. Набір протоколів, як правило, дуже обмежений. Наприклад, в перших версіях CheckPoint FireWall-1 (CheckPoint FW-1) підтримувалися протоколи Telnet, FTP, HTTP, а в Cisco Private Internet Exchange (Cisco PIX) - FTP, HTTP, H.323, RSH, SMTP і SQLNET.[13] Згодом дані набори незначно розширювалися. Також відомо, що дана технологія використовується в продуктах компаній McAfee і Symantec. Міжмережеві екрани, що використовують цю технологію, відносяться до другого покоління. Дана технологія більш гнучка в порівнянні з SPI і, крім розмежування доступу, підходить для більшого числа завдань - кешування вмісту, аналіз стисненого / шифрованого трафіку, обмеження функціонала окремих протоколів шляхом заборони окремих команд. Завдяки підключенню в режимі проксі, може служити в якості Wan Optimizer'а.

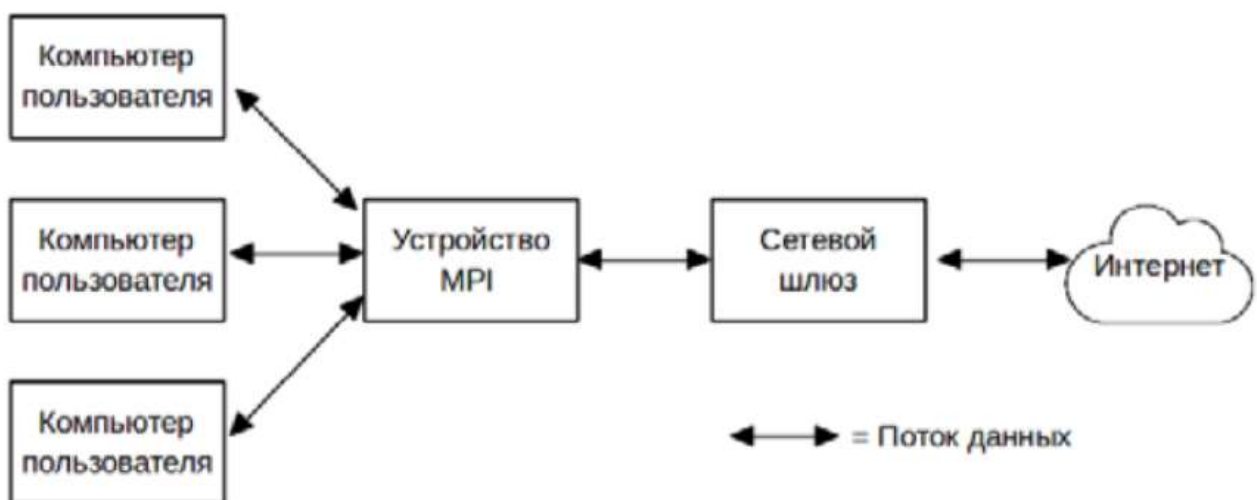


Рис. 1.2. Схема застосування пристроїв аналізу на основі технології MPI

Основний недолік MPI - погана масштабованість: кожна команда і протокол вимагають окремого «шлюзу» (вхідний-вихідний порти). Крім того, робота в режимі проксі сильно знижує швидкість обробки. Для зниження навантаження на проксі-сервер був розроблений протокол ICAP, що дозволяє проксі-серверам відправляти через них дані для проведення аналізу стороннім серверам на предмет безпеки або аналізу вмісту. Ця схема реалізована в

антивірусному продукті ClamAV, який може підключатися до проксі-серверів Squid і NetCache [19]. Ці фактори сильно обмежують застосування даної технології на рівні провайдерів інтернету внаслідок необхідності аналізу великого числа протоколів і команд на широких каналах зв'язку.

Іноді вживають більш вузький термін - DPP (Deep Packet Processing), який має на увазі такі дії над пакетами, як модифікація, фільтрація або перенаправлення. Сьогодні обидва терміни часто використовуються як взаємозамінні. Дана технологія є логічним розвитком MPI. В рамках даного підходу аналізатор переглядає вміст кожного пакета повністю. Одним з важливих відмінностей від попередніх технологій є те, що системи на базі DPI можуть приймати рішення не тільки по вмісту пакетів, але і за непрямими ознаками, властивим якимось певним мережевим програмами і протоколам. Для цього може використовуватися статистичний аналіз.

Наприклад, аналіз частоти зустрічі певних символів, довжин пакетів, відстань між мітками часу послідовних пакетів і т.д. Також, в порівнянні з попередніми підходами, значно розширено список застосувань технології: класифікація, обмеження смуги, пріоритезація, маркування, кешування і т. Д. Технологія DPI отримала розвиток, перш за все, через стрімке зростання обчислювальних здібностей процесорів, їх швидкодії і, відповідно, можливостей для більш повного і точного аналізу мережових даних [19].

На відміну від MPI, дана технологія спочатку розроблялася для високошвидкісної обробки та ідентифікації великої кількості додатків в реальному часі. Таким чином, рішення на основі DPI добре масштабуються як по ширині мережевого каналу (відомі рішення, що працюють на каналах близько 100 Гбіт / сек), так і за кількістю ідентифікованих додатків (в існуючих рішеннях - порядку декількох тисяч). З точки зору реалізації, основний компонент будь-якого рішення DPI - модуль класифікації, що відповідає за класифікацію мережових потоків. При цьому в залежності від цілей застосування DPI, класифікація може виконуватися з різною точністю:

- тип протоколу або додатка (наприклад, Web, P2P, VoIP);

- конкретний протокол рівня додатка (HTTP, BitTorrent, SIP);
- додаток, що використовує протокол (Google Chrome, μ Torrent, Skype).

Важливо відзначити, що відповідність між класами різних рівнів точності неоднозначно, що показано на рисунку 1.3.

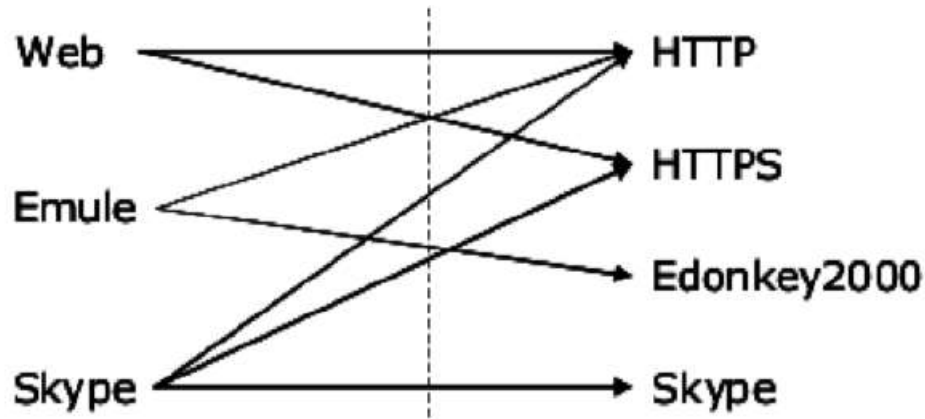


Рис. 1.3. Різниця між ідентифікацією додатків (зліва) і протоколів (праворуч).

1.3. Особливості передачі мережевого трафіку

Кожен мережевий пакет складається з керуючої інформації і корисного навантаження. Тут і далі термін «пакет» застосовується в якості універсального для узагальнення таких понять, як фрейм, дейтаграмма, сегмент відповідних мережевих протоколів. В процесі розбору в пакеті виділяються заголовки протоколів, аналізуються значення полів в них. Структура заголовка визначається специфікацією, тоді як корисне навантаження може містити довільним чином організовані дані, хоча зазвичай являє собою пакет протоколу наступного, більш високого рівня: для продовження розбору необхідно визначати, який це протокол (рис. 1.4.).

Відповідно до моделі OSI заголовки мережевих протоколів пакета утворюють стек і, як правило, слідують один за одним у природному порядку - від низького рівня до високого. Однак при організації тунельних з'єднань цей порядок може бути порушений - наприклад, при передачі IPv4-пакетів (мережевий рівень) в рамках пакетів протоколу UDP [20] (транспортний

рівень). Тунельні протоколи в даний час набули широкого поширення: зокрема, вони використовуються при організації віртуальних приватних мереж. [17] У загальному випадку можлива побудова тунелю довільної конфігурації: зокрема, один тунель може бути вкладений в інший. Розбір тунельного трафіку повинен підтримуватися мережевим аналізатором.

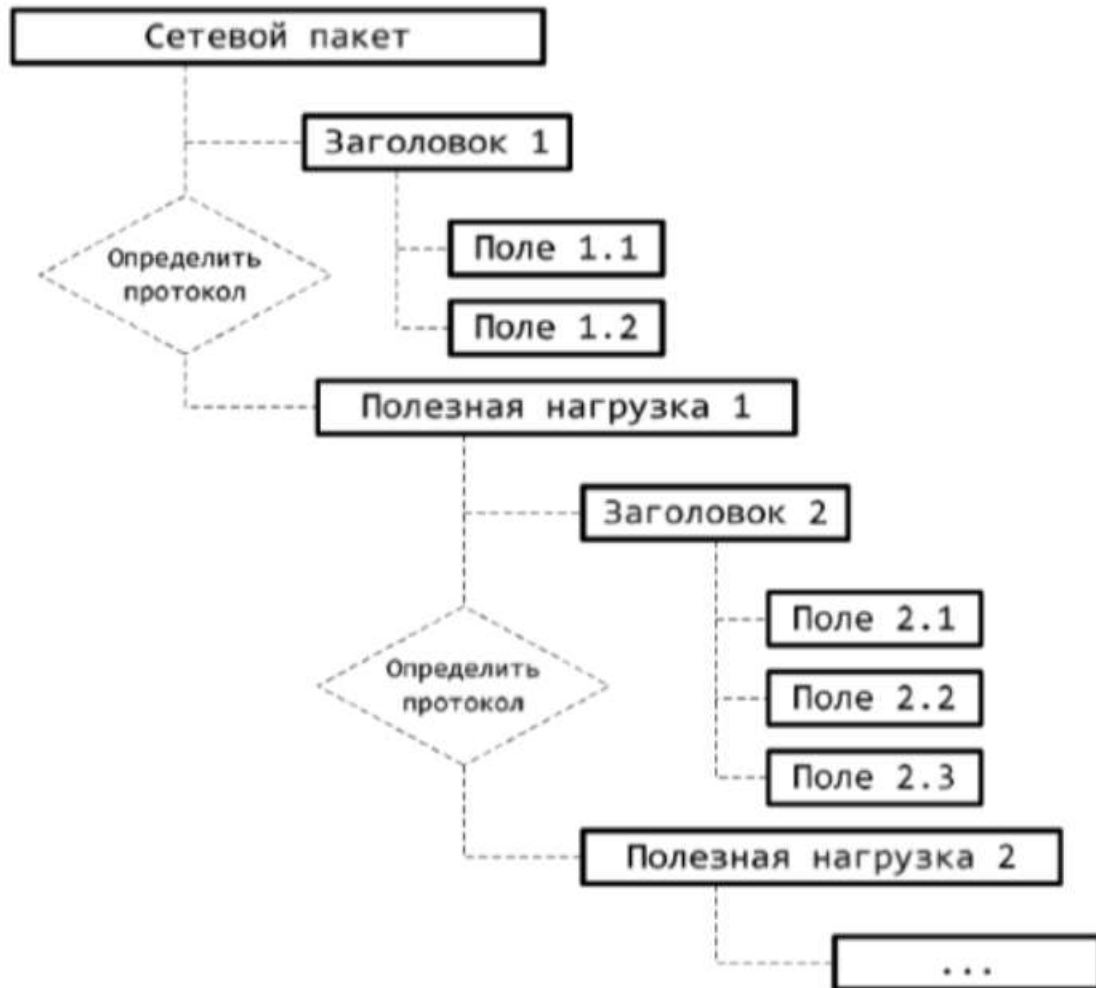


Рис. 1.4. Виділення та розбір заголовків протоколів в пакеті

Виділяють протоколи зі збереженням і без збереження стану. Обов'язковою частиною специфікації протоколу зі збереженням стану є відповідний автомат станів (Protocol State Machine). При проведенні аналізу в режимі реального часу кількість з'єднань, для яких необхідне збереження характеристик поточного стану, може необмежено рости. Тому аналізатор повинен гнучко управляти розподілом доступних йому ресурсів.

Важливою характеристикою мережевого протоколу є MTU (Maximum Transmission Unit) - максимальний розмір даних, які можуть бути передані в рамках одного пакету. Для протоколу IPv4 значення MTU складає 65535. Оскільки на практиці IPv4-пакети зазвичай інкапсулюються в Ethernet-фрейми, результуюче значення MTU визначається відповідно до конкретної версії стандарту Ethernet [21], підтримуваного мережевим обладнанням. Для блоків даних з розміром, що перевищує MTU, проводиться фрагментація: відправник розбиває блок на порції допустимого розміру, після чого кожна порція передається в рамках окремого пакета. Одержувач, таким чином, повинен виконати дефрагментацію: відновити вихідний блок з отриманих окремо порцій. Для протоколу IPv4 останній фрагмент визначається скинутим прапором MF (More Fragments): при цьому в ньому не містяться (Рисунок 1.5.) дані наступної PDU (Protocol Data Unit - одиниця передачі).



Рис. 1.5. Приклад фрагментації IPv4

У випадку протоколу TCP (рис. 1.6.) неформальною ознакою «останнього» для заданого PDU сегмента є PSH-прапор, однак цей сегмент в загальному випадку містить дані наступної одиниці передачі звідси виникає завдання визначення меж.

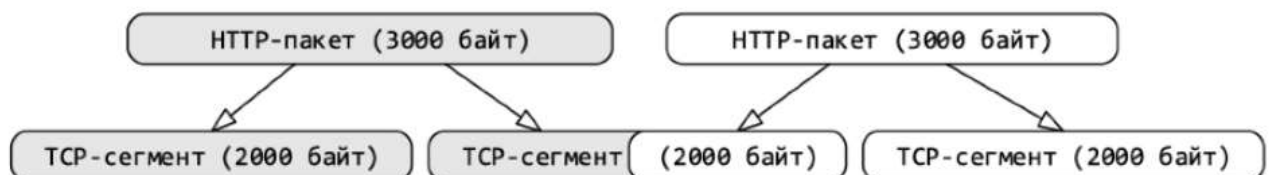


Рис. 1.6. Приклад сегментації TCP

Таким чином для проведення глибокого аналізу потрібно:

- відновлювати початковий порядок пакетів;

- визначати грані вище за списком PDU.

Питання з упорядкування пакетів для протоколів IP та TCP розглядаються в [22, 23].

Для забезпечення безпеки з'єднань деякі протоколи передбачають передачу даних в зашифрованому виді наприклад сімейство протоколів TLS [24, 25]. Щоб проаналізувати зашифровані дані, необхідно до того їх розшифрувати, використав наданий користувачем ключ(Рисунок 1.7.): аналізатор повинен мати інтерфейс для додавання відсутньої для проведення розбору інформації.

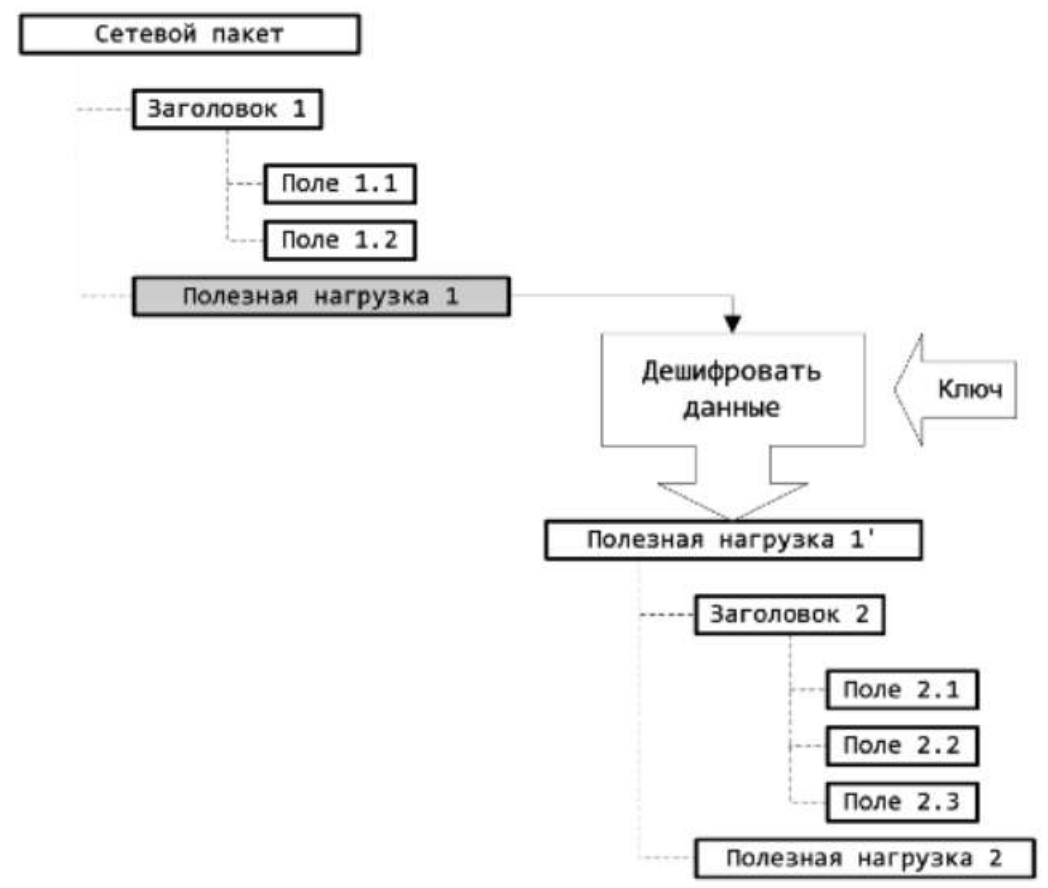


Рис. 1.7. Розшифровка даних з використанням зовнішнього ключа

При аналізі трафіку неминуче виникають помилки розбору. Під цією помилкою розуміється невідповідність між специфікацією протоколу(кодом який виконує розбір) та даними, розбір яких проводиться згідно цієї специфікації. Причини виникнення помилок розбору різняться:

- недокументовані можливості протоколу;

- викривлення даних при транспортуванні мережею;
- помилки в коді аналізатора;

Помилки розбору повинні легко локалізуватися та відтворюватися. Якщо помилка, яка виникла не є критичною, аналіз повинен продовжуватися.

РОЗДІЛ 2. ЗАСОБИ МОНІТОРИНГУ МЕРЕЖЕВОГО ТРАФІКУ

2.1. Пакетні сніфери

Основним інструментом для спостереження повідомлень, якими обмінюються хости є сніфер пакетів. Як випливає з назви, сніфер пакетів захоплює ("нюхає") повідомлення, передані з комп'ютера або отримані ним. Крім цього програма-сніфер, як правило зберігає та відображає вміст різних полів протоколів у цих захоплених повідомленнях. Сніфер пакетів – це пасивна програма. Вона досліджує повідомлення, що відправляються і одержуються програмами і протоколами, які працюють на комп'ютері, але ніколи не відправляє пакети сама. Отримані пакети ніколи не адресуються сніферу. Сніфер пакетів отримує копію пакетів, переданих/отриманих додатками або протоколами, які виконуються на комп'ютері.

Пакетний сніфер — це або програмний, або апаратний інструмент для перехоплення, реєстрації та аналізу мережевого трафіку та даних. Ці інструменти допомагають визначити, класифікувати та усунути неполадки мережевого трафіку за типом програми, джерелом та пунктом призначення. На ринку є різноманітні інструменти, більшість з яких покладаються на інтерфейси прикладних програм (API), відомі як `pcap` (для Unix-подібних систем) або `libpcap` (для систем Windows) для захоплення мережевого трафіку. Тоді найкращі сніфери пакетів аналізують ці дані, що дозволяє вам точно визначити джерело проблеми та не допустити її в майбутньому[1].

Щоб по-справжньому зрозуміти важливість сніфферів , важливо розуміти як відбувається маршрутизація в Інтернеті. Почнемо з початку. Кожен електронний лист, який ви надсилаєте, відкрита веб-сторінка та файл, яким ви ділитесь, поширюється в Інтернеті як тисячі маленьких керованих фрагментів, відомих як пакети даних. Ці пакети передаються через стек протоколів, відомий як протокол управління передачею /

протокол Інтернету (TCP / IP). TCP / IP розбивається на чотири шари: рівень протоколу додатків, рівень протоколу управління передачею (TCP), рівень інтернет-протоколу (IP) та апаратний рівень [2].

Кожен пакет переміщується через рівень програми вашої мережі до рівня TCP, де йому присвоєний номер порту. Далі, пакет переходить на IP-рівень і отримує свою цільову IP-адресу. Як тільки пакет має номер порту та IP-адресу, він може бути відправлений через Інтернет. Надсилання здійснюється через апаратний рівень, який перетворює пакетні дані в мережеві сигнали. Коли пакет прибуває до місця призначення, дані, які використовуються для маршрутизації пакету (номер порту, IP-адреса тощо), видаляються, і пакет рухається далі через стек протоколів нової мережі. Після досягнення вершини він збирається в первісну форму.

Пакетні сніфери працюють, перехоплюючи дані про трафік під час проходження по дротовій або бездротовій мережі та копіюючи їх у файл. Це відомо як захоплення пакетів. Хоча комп'ютери, як правило, розроблені для того, щоб ігнорувати ступінь трафікової активності від інших комп'ютерів, пакетні сніфери це перетворюють. При встановленні програмного забезпечення, мережева карта інтерфейсу (NIC) - інтерфейс між вашим комп'ютером та мережею - повинна бути встановлена в розрядний режим. Це дає змогу комп'ютеру зафіксувати та обробити через sniffer пакет все, що потрапляє в мережу [3].

Що можна захопити, залежить від типу мережі. Для дротових мереж конфігурація мережевих комутаторів, які відповідають за централізацію зв'язку з декількох підключених пристроїв, визначає, чи може мережевий сніфер бачити трафік у всій мережі або лише на її частині. У бездротових мережах інструменти збору пакетів зазвичай можуть захоплювати лише один канал одночасно, якщо хост-комп'ютер не має безлічі бездротових інтерфейсів.

Отже, у чому сенс аналізаторів пакетів? Сніффер може допомогти вам орієнтуватися на нові ресурси при розширенні пропускнуєї спроможності

мережі, керуванні пропускною здатністю, підвищенням ефективності, забезпеченням ділових послуг, підвищенням безпеки та покращенням роботи кінцевих користувачів. Що стосується великих та малих компаній, щоденні завдання можуть бути негайно зірвані проблемами ефективності, пов'язаними з мережею, додатком чи обома. Щоб відновити роботу їхньої компанії, систематики повинні мати можливість швидко визначити першопричину. Оскільки sniffers пакетів переглядають та збирають інформацію для всього трафіку по всій мережі, вони можуть оцінювати критичні шляхи мережі, щоб допомогти адміністраторам визначити, що програма чи мережа є причиною поганого досвіду користувачів. З цією інформацією, адміністратори краще оснащені для визначення та вирішення походження проблеми.

Коли користувачі повідомляють про повільність, адміністратори можуть використовувати аналіз PCAP для вимірювання часу реакції в мережі - також відомий як затримка мережевого шляху - та визначати кількість часу, необхідного для переходу пакета через мережевий шлях від відправника до одержувача. Це дозволяє адміністраторам швидко визначити причину уповільнення та виявити постраждалі програми, щоб вжити заходів.

Проаналізуйте трафік за типом. Оцінюючи проблеми з мережею та додатками, першочергове значення має трафік у вашій мережі. За допомогою правильного аналізатора IP-sniffer та пакетів трафік класифікується на типи на основі IP-адрес сервера призначення, використовуваних портів та вимірювання загального та відносного обсягу трафіку для кожного типу. Це дає вам змогу виявити надмірний рівень некомерційного трафіку (наприклад, соціальних медіа та зовнішнього веб-серфінгу). Ви також можете визначити трафік, що протікає через мережеве посилення, а також трафік на конкретні сервери або програми для цілей управління ємністю.

Поліпшення пропускнуої здатності. Коли користувачі скаржаться на те, що «мережа повільна» або «Інтернет знижується», продуктивність припиняється, знижуючи рентабельність інвестицій . Щоб виправити цю помилку , вам потрібно зрозуміти, як та ким використовується пропускна здатність мережі. Сніфер пакету Wi-Fi може отримати показники продуктивності для автономних точок доступу, бездротових контролерів та клієнтів. Багато з них також пропонують моніторинг несправностей, продуктивності та доступності мережі, кореляцію даних між стековим стеком, аналіз мережевого шляху та багато іншого, щоб допомогти вам виявити потенційні проблеми та мінімізувати час простою мережі. Поліпшення безпеки. Великий обсяг вихідного трафіку може означати, що хакер використовує ваші програми, або для спілкування зовні, або для передачі великої кількості даних.

2.2.Огляд інструментів обробки пакетів

Сьогодні на ринку є незліченна кількість інструментів, як платних, так і безкоштовних. І хоча кожен інструмент побудований на основних принципах збору мережевого трафіку, вони значно відрізняються за своєю шириною та глибиною. Багато інструментів з відкритим кодом є надзвичайно простими у своїй конструкції, і в цьому справа: ці інструменти створені для забезпечення надійного, чистого збору даних, залишаючи якнайменший слід. Якщо вам потрібні прості сніфери та швидка діагностика, безкоштовний інструмент з відкритим кодом може бути корисним. Багато - хоча і не всі - безкоштовні версії можна оновити, щоб надати додаткові аналітичні функції, якщо потрібна більша підтримка.

З такою кількістю продуктів на ринку важко дізнатися, який сніферний пакет вибрати. Незважаючи на те, що безкоштовних варіантів вистачає, покупка сніффера пакета може гарантувати, що ви озброєні

інструментом, який не тільки фіксує дані, але й пропонує інтуїтивний аналіз. Виходячи за рамки ваших базових сніферів, яких налічується десятки, ви знайдете більш надійні аналітичні інструменти для збору пакетів та мережевого збору даних. У багатьох випадках, що відрізняє ці інструменти, - це здатність проводити глибоку перевірку пакетів (DPI).

Ці великі інструменти на рівні підприємств часто оснащені для оповіщення про випадки виключення та створення інтуїтивно зрозумілих графіків та діаграм із відображенням детальних показників. Хоча вони йдуть високою ціною, вони варті своїх інвестицій.

TCPDUMP - це популярний інтерфейс командного рядка (CLI) та інструмент дослідження пакетів з відкритим кодом, сумісний на платформах Unix та Linux. Він був винайдений у 1987 році в Національній лабораторії Лоуренса Берклі, а після цього опублікований через кілька років. У ньому є бібліотека libpcap, розроблена мовою програмування на C, яка працює для збору інформації мережі. Libpcap забезпечує інтерфейс для всіх загальних платформ на базі Unix, включаючи FreeBSD та Linux. Інтерфейс libpcap в платформі Windows під назвою WinDump. Windump використовується WinPcap, який є портом Windows бібліотеки libpcap. Розробники розробляли бібліотеку libpcap як API незалежної платформи для роботи над різними програмами та усунення системної залежності для модулів збору даних у кожній програмі. TCPCDump розглядається як інструмент розбору [5].

За замовчуванням він перехоплює та друкує пакет, який захопило з мережі; інші функції, такі як зберігання, виконуються заданими командами. TCPCDump працює так:

- 1) Читання / запис захопленого файлу з мережі в пакеті CAPture (PCAP) за допомогою команд CLI.
- 2) Він фільтрує пакети за деякими заданими параметрами.
- 3) Він друкує на екрані захоплені дані відповідно до заданих параметрів. Це більш легкий і портативний інструмент для сніфу пакетів,

оскільки він залежить лише від CLI, і мережеві адміністратори використовують його для доступу до мережевих пристроїв з віддаленого місця. На Рисунок 1 показаний трафік TCP / IP та його аналіз дослідження пакетів TCPDump, відображення адреси і вмісту трафіку даних (Рис. 2.1.).

Основне обмеження TCPDump, він не надає адміністратору мережі візуального GUI захоплених даних для більшого аналізу, є тільки CLI.[9] Оскільки, це текстовий формат і користувачеві легко користуватися ним дистанційно через з'єднання Telnet. Є ще кілька недоліків у TCPDump.

```

root@yoshiki # tcpdump -i lo -x
tcpdump: listening on lo
11:17:49.511923 localhost.33882 > localhost.8765 P 1502698231:1502699255(1024
k 1504308678 win 32767 <nop,nop,timestamp 23470237 23466753> (DF)
4500 0434 5a16 4000 4006 deab 7f00 0001
7f00 0001 845a 223d 5991 5af7 59a9 edc6
8018 7fff d05d 0000 0101 080a 0166 209d
0166 1300 6865 6c6c 6f0a 0040 90b0 1440
0100 0000 0000 0000 0002 0000 44f7 ffbf
0002
11:17:49.516227 localhost.8765 > localhost.33882 P 1:1025(1024) ack 1024 win
7 <nop,nop,timestamp 23470238 23470237> (DF)
4500 0434 e524 4000 4006 539d 7f00 0001
7f00 0001 223d 845a 59a9 edc6 5991 5ef7
8018 7fff 1f9d 0000 0101 080a 0166 209a
0166 209a 4845 4c4c 4f0a 0040 90b0 1440
0100 0000 0000 0000 0002 0000 44f7 ffbf
0002
11:17:49.516271 localhost.33882 > localhost.8765: . ack 1025 win 32767 <nop,nc
timestamp 23470238 23470238> (DF)
4500 0034 5a17 4000 4006 e2aa 7f00 0001
7f00 0001 845a 223d 5991 5ef7 59a9 f1c6
8010 7fff 0a23 0000 0101 080a 0166 209e
0166 209e

3 packets received by filter
0 packets dropped by kernel

```

Рис. 2.1. Огляд TCPDump, показує потік характеристик TCP/IP

До них належать:

- Обмеження в аналізі трафіку, можуть застосовуватися лише протоколи на основі TCP.
- Він повідомляє лише те, що знаходить у пакетах, якщо IP-адреса підроблена в трафіку, вона не має можливості повідомляти нічого іншого.

- Пакети, заблоковані брандмауером, не відображаються.

Wireshark винайдений вченим Джеральдом Комбсом наприкінці 1997 року для перевірки та розпізнавання проблем мережі та моніторингу трафіку даних. Він назвав його Ethereal до травня 2006 року, після чого його назва змінилася на Wireshark. Це програмне забезпечення з відкритим кодом, безкоштовний інструмент і аналізатор пакетів для графічного інтерфейсу, який написаний мовою програмування на C та випущений під ліцензією GNU General Public License (GPL). Він працює на різних Unix-подібних операційних системах, включаючи Mac OS X, Linux, платформу Solaris, а також операційну систему Microsoft Windows. Інтерфейс командного рядка (CLI) Wireshark називається TShark, що дозволяє користувачеві працювати з ним за допомогою команд. Це як TCPDump з додатковим графічним інтерфейсом, підтримуючи різноманітні протоколи та можливість фільтрування та сортування. Він використовується в мережевому інструменті судового аналізу (NFAT) в організаціях.

Wireshark призначений для захоплення пакетів з працюючих мереж, а також для перегляду раніше збереженого файлу даних. Підтримуваний формат захоплення пакетів — це формат файлу "PCAP". Він відображає захоплені дані у байтовому та шістнадцятковому форматах, де відображаються різні типи використаних пакетів та протоколів. Це також дозволяє користувачеві збирати дані пакетів у потік TCP [8].

Він має інтерфейс з трьома панелями; панель підсумків або панель списку пакетів, яка показує різні аналізовані пакети, такі як номер кадру, дата, час, адреса призначення та джерела IP, протоколи верхнього рівня, довжина пакету та інформація вмісту трафіку з кольором для кожного захопленого типу пакету. Друга панель - це захоплені деталі пакета. Коли пакет визначається на панелі списку пакетів, деталі з'являються на наступних двох панелях; деталі та байтові або шістнадцяткові панелі. Панель деталей відображається як деревоподібна структура протоколів,

Тут показаний інтерфейс Wireshark, що відображає три вікна; панель підсумків, деталі та байт з різними характеристиками трафіку мережі в читаному вигляді. У розділі деталей видно, що номер кадру становить 60 байт, тип мережі - це Ethernet II, тип протоколу - IPv4, а вміст корисної навантаження - ARP.[10] Розмір кадру вимірюється за допомогою модуля максимальної передачі (MTU), і цей блок визначається відповідно до використовуваного типу мережі. Наприклад, MTU для асинхронного режиму передачі (ATM) становить 53 байти, MTU в мережах Ethernet і IPv4 - 1500 байт, і в інших типах мереж використовується перемикання кадрів, які досягають 9000 байт. Отже, розмір кадру змінюється відповідно до MTU у заданому типі мережі.

Крім того, розмір кадру визначають також відповідно до використовуваної програми в мережі. У цьому випадку, це 60 байтів, в яких ARP використовується для пошуку фізичної адреси інтерфейсу маршрутизатора або хоста, коли вказана його логічна IP-адреса. В іншому прикладі дані кадру розміром 500 байт за допомогою застосованої якості обслуговування (QoS) в деяких додатках охорони здоров'я на базі UDP і зроблено висновок, що він підходить як для затримки, так і для тремтіння, а також використано деякі інші QoS що призвело до отримання кадру розміром 1500 байт. Потім кадр з'являється в інструменті дослідження пакетів відповідно до програми або протоколу, який використовується в мережі.

Обмеження Wireshark, воно потребує найкращого розуміння форматів протоколів, HTTP та Cascade Style Sheet CSS, знання мови у форматі байтів. Він використовує формат файлу PCAP для фіксації трафіку, тому він може захоплювати пакети лише тих типів мережі, які підтримують формат файлу PCAP. Ще одним недоліком є те, що Wireshark не є автоматизованим інструментом, він не є підтримкою для моніторингу тривалого часу.

Colasoft - це інструмент аналізатора протоколів мережевого протоколу із закритим джерелом, призначений для роботи на платформі операційної

системи Windows, що використовується для особистого використання адміністратором мережі для усунення несправностей, моніторингу та діагностики трафіку в комп'ютерній мережі. Capsa випускає безкоштовні інструменти Colasoft, що забезпечує простоту використання, аналіз пакетів у режимі реального часу та надійний криміналістичний, поглиблений аналіз протоколів, і в нього 24-годинний моніторинг мережі. Він має особливість відкриття декількох інтерфейсів в одному екземплярі, надання користувачеві графічних інтерфейсів та матричних представлень.

Він має глибокий аналіз пакетів, що показують різні характеристики з можливістю генерування звітів, журналів та попередження лише голосовими та електронними повідомленнями для ліцензованих версій [14]. Він має різноманітні функції графічного інтерфейсу, відображаючи захоплену інформацію у графіках, матриці та відповідно до кожної характеристики трафіку мережі, що показує кожен протокол, що використовується в мережі. На рисунку 2.3 показаний інструмент Colasoft та його розширені функції GUI.

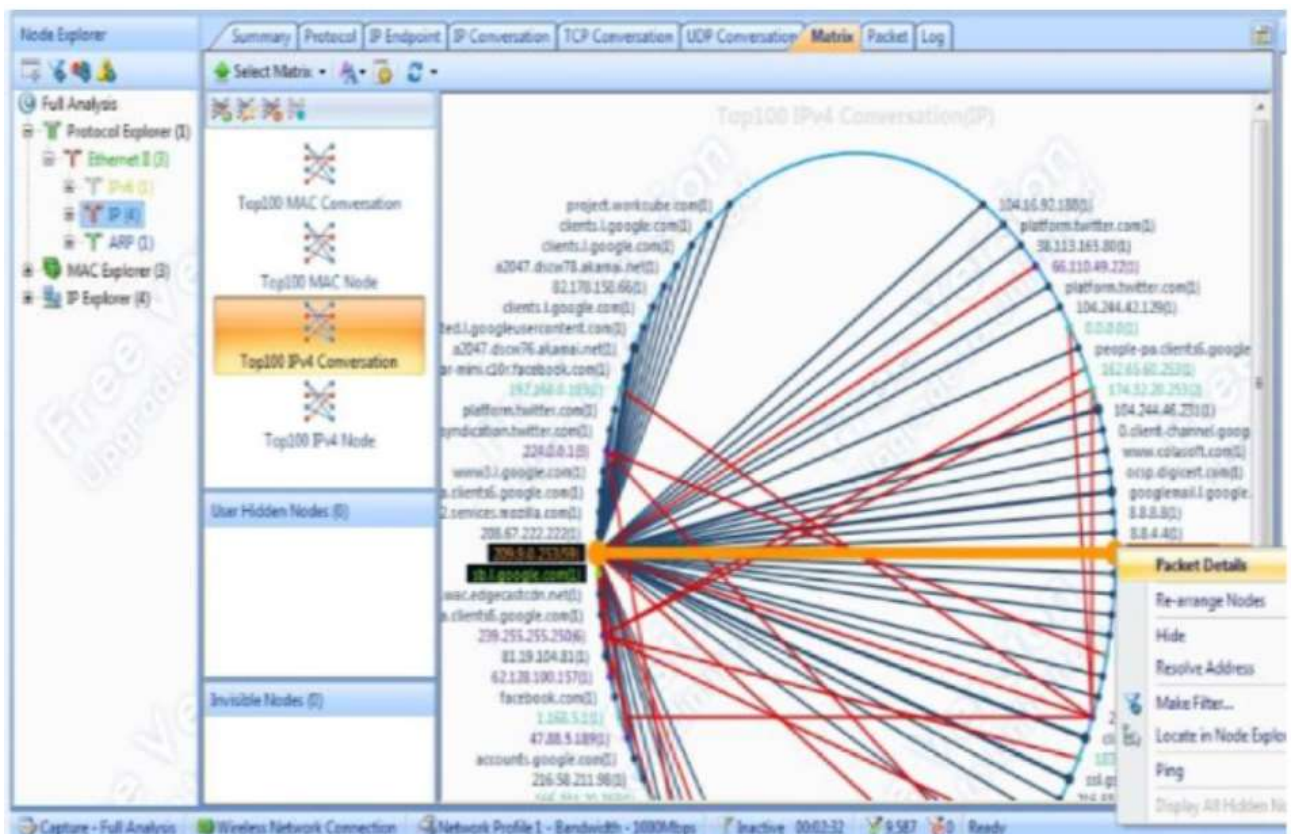


Рис. 2.3. Інтерфейс інструменту Colasoft.

Є деякі обмеження Colasoft; це дорогий додаток, тоді як безкоштовна версія доступна, але з обмеженими можливостями, наприклад, безкоштовна версія не повідомляє користувача через електронну пошту та голосові канали. Ще два недоліки інструменту Colasoft - це те, що він працює лише на платформі операційної системи Microsoft Windows, і він підтримує лише 300 протоколів, що вважається меншим порівняно з іншими інструментами для пакетів, такими як Wireshark.

2.3. Порівняльний аналіз інструментів обробки пакетів

Для порівняння між перерахованими вище трьома методами мережевого аналізу ми залежимо від таких параметрів, як код з відкритим вихідним кодом, кількість підтримуваних протоколів, підтримувана операційна система, підтримує PCAP, користувальницький інтерфейс, вартість, форми декодування, визначені аномальні пакети, реконструюють потік TCP, тощо.

В таблиці 2.1 показано порівняння між інструментами Wireshark, TCPDump та Colasoft. З таблиці порівняння, жоден інструмент аналізу пакетів не веде його за всіма параметрами. Але порівняння з перевагами та недоліками допоможе розробникам вдосконалити інструмент дослідження пакетів, щоб подолати ці обмеження. Тут ми порівняємо інструменти, використовуючи якісні та кількісні параметри.

Таблиця 2.1

Порівняльні характеристики між Wireshark, TCPDump та Colasoft
Packet Sniffing Tools

Параметри	Сніферні інструменти		
	TCPDump	WireShark	Colasoft
Відкритий код	+	+	-
Якими операційними системами підтримується	Linux(WINDum для Windows)	Linux, Windows	Windows
Число підтримки протоколів	TCP/IP	Більш ніж 300	300
Користувацький інтерфейс	CLI	CLI і GUI	GUI
Вартість	Безкоштовно	Безкоштовно	999 \$
Libpcap основа	+	+	-
Визначення	-	+	+

прихований даних			
Використання місця на диску	484КБ	449МБ для Unix 89МБ для Windows	32МБ
Відображення в додатку шару протоколу	-	+	+
Декодування протоколу	Лише Hex, ASCII	Лише Hex, ASCII	EBDIC, Hex, ASCII
Відновлення TCP потоку	-	+(але лише форматоване)	+
Виявлення ненормальних даних	-	-(лише створює попередження)	+
Кілька інтерфейсів	-	-	+
Сповіщення знаходження	-	-	+
Відновлення HTTP веб сторінки	-	-(показує актуальний контент трафіку індивідуально)	-(показує лінки контенту трафіка індивідуально)
Мережева комунікаційна матрична мапа	-	-	+
Оцінка критичного та некритичного для бізнесу трафіку	-	+(за допомогою створення нових фільтрів та пошуку)	+(вбудована)
Можливість розробляти та налаштовувати розробниками	+(можливе налаштування користувачем під себе)	+	-(лише командою розробки capsa.co)
UDP трафік	-	+	+

Colasoft має функції аналізу більш візуального пояснення зі статистикою захоплених пакетів, відображенням більшої кількості інформації про протоколи та користувацькі програми з графіками та матричним поданням для всіх підключених кінцевих точок. Додаткові функції, які Colasoft включає в себе звіти, журнали аудиту та діагностику. Усі ці функції допомагають адміністраторові мережі діагностувати проблеми мережі. Також Colasoft має потужний аналіз та інтерпретацію потоку TCP; він має універсальну пропускну здатність, мережевий трафік та аналіз використання [21]. Він має матричне подання та функції поглибленого декодування трафіку з багаторазовою поведінкою мережевого моніторингу.

Крім того, вона має візуалізацію приховування комп'ютерних мереж. Крім того, Colasoft має більш потужну видимість та стиль Windows 7 з простими графічними екранами, інформаційними панелями та мережевим аналізатором. За допомогою цього інструменту для користувача легко і просто зробити аналіз ,просто завдання, яке ви хочете, виконується натисканням миші. Отже, Colasoft стає більш зручним інструментом для дослідження пакетів, він забезпечує легкий для читання спосіб і має декілька інтерфейсів в одному екземплярі. Графіки показують більшу візуалізацію для різних статистичних даних і властивостей мережі. Wireshark обмежений цими можливостями GUI, і він не відкриває декілька інтерфейсів в одному екземплярі.

У порівнянні з Wireshark, Colasoft забезпечує більшу мережеву безпеку за допомогою сповіщень про попередження через аудіо та електронні листи. Недолік Colasoft, він охоплює лише 300 протоколів, що дуже менше порівняно з Wireshark, який підтримує 1100 протоколів.

TCPDump - це дуже портативний і економічний пакет нюхає інструмент з точки зору використання пам'яті, оскільки він займає лише 484 КБ місця для встановлення. Хоча розмір інсталяційного файлу Wireshark на початку інсталяції становить 18 МБ, але після завершення інсталяції він споживає 81 Мбайт у Windows та 449 МБ дискового простору в операційній системі Linux. Місце для встановлення Colasoft становить 32 Мб. Отже, що стосується використання пам'яті, то Wireshark дуже дорогий.

Оскільки Wireshark - це відкритий код, кожен може завантажити його код і вдосконалити його. У світі існує багато універсальних розробників, які мають можливість налаштування та вдосконалення цього інструменту, в той час як Colasoft обмежений лише командою розробників компанії Capsa. Отже, Wireshark вважається хорошим інструментом дослідження пакетів для розуміння функцій програмування, і він відповідає вимогам користувачів мережі, здійснюючи налаштування без витрат. У інструменті Colasoft, якщо користувачеві потрібна певна налаштування для певної

проблеми моніторингу мережі, він вимагає від компанії, яка надає оплату, за ці налаштування. За допомогою інструменту аналізу пакетів Wireshark ви можете отримати більше досвіду конфігурації TCP / IP, розуміючи структуру мережі, а також вона працює на різних платформах, включаючи Linux, Solaris, OS X і Windows.

Крім того, деякі автори проводять дослідження в цій галузі для вдосконалення інструменту Wireshark, тут вдосконалено інструмент дослідження пакетів Wireshark для виявлення вторгнення типів атак Denial of Service (DoS), особливо в тому випадку, коли можна подолати атаку затоплення ping, яка надсилає велику кількість команд ping на пристрій жертви.

Усі ці інструменти мають загальні характеристики мережесовместимостей, але кожен інструмент має свою конкурентну особливість. Існують різноманітні якісні та кількісні параметри, які обговорюються та порівнюються на інструментах: Wireshark, TCPDump і Colasoft. З цих параметрів - кількість підтримуваних протоколів, відкритий вихідний код, платформа, що підтримується, бібліотека libpcap, підтримка PCAP, користувацький інтерфейс, вартість, форми декодування, визначені аномальні пакети, мережесовместимий зв'язок у матричній карті, реконструкція потоку TCP тощо.

Отже, кожен інструмент мережесовместимого аналізатора не призводить до всіх мережесовместимих параметрів. Оскільки інструмент Colasoft кращий за Wireshark у матричних та графічних звітах, Wireshark - це відкритий вихідний код, який легко розробляти та налаштовувати код всім відповідно до їх потреб, і він сумісний з різними платформами, такими як Linux та Операційні системи MS Windows, а Colasoft працює лише на операційних системах MS. З іншого боку, інструмент TCPDump - це легкий інструмент, який займає невеликий розмір простору і це конкурентна особливість, тому найкращим варіантом його дистанційного використання для моніторингу комп'ютерних мереж за допомогою

інтерфейсу командного рядка. Іншим важливим фактором параметра є кількість протоколів, що підтримуються засобом сніферів пакетів. Wireshark підтримує величезну кількість протоколів понад 1000 протоколів, що є чудовим інструментом для моніторингу та контролю мереж, що використовується в різномірних мережах, які мають різноманітні протоколи, включаючи відео та аудіо програми, в той час як інші інструменти мережевого моніторингу підтримують лише декілька таких протоколів оскільки інструмент Colasoft підтримує близько 300 протоколів, а TCPDump підтримує протокол TCP / IP і не підтримує протокол транспортного рівня протоколу користувача (UDP).

Крім того, враховуючи інші параметри, такі як вартість, Wireshark і TCPDump є безкоштовними інструментами, в той час як Colasoft є дорогим і дорожчим, ніж інші інструменти. Але Colasoft є більш сильним інструментом для виявлення ненормальних протоколів, що є конкурентною особливістю порівняно з іншими інструментами, такими як Wireshark, які лише попереджають. Інструмент Colasoft розроблений командою Capsa, який став хорошим графічним інтерфейсом і має більше функцій безпеки. Інтерфейс фільтрації також розглядається як конкурентна особливість в інструменті Colasoft, який є з графічним інтерфейсом та зручним для користувача, що дозволяє користувачеві легко фільтрувати та аналізувати протоколи та трафік даних. В таблиці 2.2 показано найкраще використання інструменту дослідження пакетів для кожного ресурсу мережі.

Таблиця 2.2

Найкраще використання інструментів обробки пакетів; Інструменти Colasoft, Wireshark та TCPDump.

Мережеві параметри	Сніферні інструменти
--------------------	----------------------

Захист комп'ютерних мереж	Colasoft
GUI	
Виявлення нетипових пакетів	
Мережеві сповіщення	
Розмір пакетів	
Мережеве спілкування	
Кілька або один інтерфейс	
Декодування протоколів Hex, ASCII, EBDIC	
Виявлення ненормальних пакетів	
Визначення пакетів з підробленими даними	
Відображення шарів протоколів додатку у OSI 7 моделі	
Підтримка OS	Wireshark
Налаштування і розробка всіма розробниками	
Час відповіді	
Пропускна здатність	
Швидкість обробки	
Число протоколів, що підтримуються	
Портативний і простий контроль віддаленого доступу	TCPDump

Найкраще використовувати Colasoft для сигналів про ненормальні та підроблені пакети, це забезпечує більшу безпеку та інтерфейси GUI. Хоча Wireshark підходить для навчання програмістами та розробниками, завантажуючи вихідний код і налаштовують його відповідно до потреб мережевого моніторингу. Інструмент TCPDump більше підходить для віддаленого логічного контролю доступу для моніторингу мережі за допомогою CLI.

Крім того, деякі інші автори перевірили деякі мережеві параметри для порівняння між засобами дослідження пакетів; результат показаний у наступних пунктах [22].

- Час відповіді Він визначається як тривалість періодів часу (вимірюється у одиницях часу) для певної конкретної події. Автори роблять висновок, що час реакції Wireshark менше часу реакції Colasoft.

- Пакети за секунди (PPS) Це кількість переданих пакетів за одну секунду. Добре видно, що Wireshark має менші втрати пакетів, ніж Colasoft. Отже, Wireshark вважається кращим порівняно з Colasoft для повторної передачі пакетів.

- Розподіл розміру пакету Менший розмір пакетів може призвести до меншої напруги в мережі, тоді як великий розмір пакетів збільшує навантаження на мережу. Після експерименту вони дійшли висновку, що розмір пакету довжини у Wireshark становить 558,76 байт, тоді як у Colasoft – 434. Отже, Colasoft надсилає довжину пакету середнього розміру; це краще, ніж інструмент Wireshark для завантаження комп'ютерної мережі. Colasoft Capsa не підкреслює систему та мережу.

- Пропускна здатність (біт на секунду) Це обсяг даних, оброблений системою, виміряний у секунду. Після експерименту показано, що пропускна здатність у Colasoft Capsa є великим діапазоном і він швидко змінюється. Ці випадкові зміни погано впливають на систему мережі, оскільки вони заважають роботі системи та комп'ютерної мережі. Тоді як Wireshark має системність та хороший діапазон моделей мережі комп'ютера. Середній біт на секунду (bps) у Wireshark становить 115,398 кбіт / с, а в Colasoft - 6,34 кбіт / с. Отже, Wireshark має більшу пропускну здатність більше, ніж Colasoft Capsa, тоді Wireshark має більшу продуктивність з постійними коливаннями, а також, не бачачи високих відсічок Bps.

2.4.FATT(Fingerprint all the things)

На основі попереднього аналізу існуючих рішень з точки зору функціоналу пакетного сніфера було створено власний — FATT(Fingerprint all the things). Розроблено скрипт для вилучення метаданих та відбитків (програмний код див. додаток Б). Загальний опис створеного продукту:

- Розроблене рішення є кросплатформним(Linux, macOS та Windows).
- З можливістю вилучення метаданих та цифрових відбитків з різних джерел трафіку з поточного та файлу пакетних даних мережі(.pcap).

- Реалізована підтримка протоколів SSL/TLS, SSH, RDP, HTTP. Для вилучення цифрових відбитків пристроїв було обрано такі методи реалізації для відповідних протоколів:

1. JA3 для протоколу TLS. JA3 — це метод створення відбитків SSL/TLS, який повинен бути легким для інтеграції на будь-якій платформі і може бути легко застосований для дослідження загроз [15,16]. Це набагато ефективніший спосіб виявлення зловмисної активності через SSL, ніж індикатори компрометації на основі IP або домену. Оскільки JA3 виявляє клієнтську програму, не має значення, чи зловмисне програмне забезпечення використовує DGA (алгоритми генерації домену) або різні IP-адреси для кожного хоста, JA3 може виявити саме шкідливе програмне забезпечення на основі того, як проходить взаємодія, а не на тому, які компоненти взаємодіють. JA3 також є чудовим механізмом виявлення в замкнених середовищах, де дозволено встановлювати лише кілька конкретних програм. У таких типах середовищ можна скласти білий список очікуваних програм, а потім попередити про будь-які інші звернення JA3.

2. HASSH для протоколу SSH "HASSH" - це мережевий стандарт відбитків, який може бути використаний для ідентифікації конкретних реалізацій клієнта та сервера SSH. Відбитки можна легко зберігати, шукати та ділитися ними у вигляді відбитка MD5. "hassh" і "hasshServer" - хеші MD5, побудовані з певного набору алгоритмів, які підтримуються різними SSH-клієнтськими та серверними програмами. Ці алгоритми обмінюються після початкового тристороннього handshake-у TCP як пакети з чітким текстом, відомі як повідомлення "SSH_MSG_KEXINIT", і є невід'ємною частиною налаштування остаточного зашифрованого каналу SSH. Існування та впорядкування цих алгоритмів є досить унікальним, щоб його можна було використовувати як відбиток, щоб допомогти визначити базовий додаток для клієнтів і серверів або унікальну реалізацію,

незалежно від нібито ідентифікаторів вищого рівня, таких як рядки "Клієнт" або «Сервер» (рис. 2.4).

3. RDP для протоколу RDP. RDP - експериментально розроблений RDP відбиток для стандартного RDP протоколу(для інших варіацій безпеки RDP-протоколу використовується TLS, таким чином відбиток може бути захоплений за допомогою JA3).

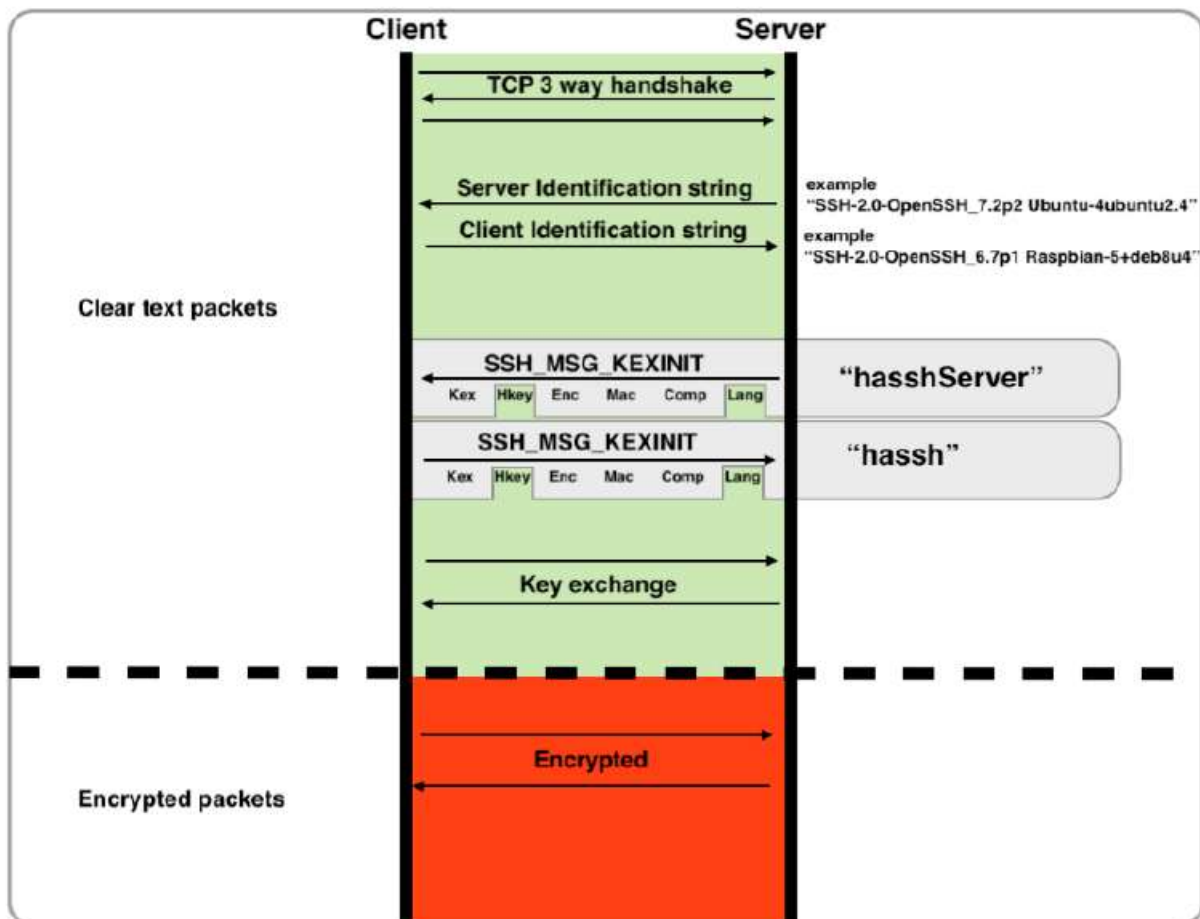


Рис. 2.4. Принцип роботи HASSH

Використання Основним варіантом використання даного FATT це моніторинг приманок (honeypots), також продукт можна використовувати для таких цілей, які перед нами ставить мережевий криміналістичний аналіз. Як користуватися fatt: usage:

```
fatt.py [-h] [-r READ_FILE]
```

```
[-d READ_DIRECTORY]
```

```
[-i INTERFACE] [-fp [{tls,ssh,rdp,http,gquic}][{tls,ssh,rdp,http,gquic} ...]]
```

[-da DECODE_AS]

[-f BPF_FILTER]

[-j][-o OUTPUT_FILE]

[-w WRITE_PCAP] [-p]

Необов'язкові аргументи:

- r, --read_file — ім'я файлу пакетних даних мережі(.pcap)
- d, --read_directory — директорія файлу пакетних даних мережі(.pcap);
- i, --interface — інтерфейс, який досліджується;
- fp, --fingerprint — види протоколів, які включені(за замовчуванням - всі);
- da, --decode_as — словник типу {decode_criterion_string: decode_as_protocol}, який використовується для того, щоб повідомити tshark як розшифрувати протоколи в незвичайних ситуаціях;
- f, --bpf_filter — BPF фільтр(лише для режиму поточного трафіку);
- j, --json_logging — формат виведення JSON;
- o, --output_file — файл журналу(за замовчуванням fatt.log);
- w --write_pcap — запис захоплених пакетів з поточного трафіку до файлу пакетних даних мережі;
- p, --print_output — виведення вихідних даних.

Результат роботи програми при аналізі поточного трафіку (див. Додаток В). Відповідний розширений JSON файл (див. Додаток Г).

Результати роботи FATT співпали з відповідними у Wireshark - з цього можна зробити висновок про правильність реалізації підтримки відповідних протоколів.

Розглянемо приклади роботи для файлів пакетних даних мережі(.pcap):

- для вразливості CVE-2020-0708 RDP (BlueKeep) (див. Додаток Д).
- перевірка за допомогою іншої вразливості CVE-2020-0708 PoC (див. Додаток Е). Цього разу ми не бачимо повідомлення RDP ClientInfo, оскільки

РoC використовує TLS (не стандартний протокол безпеки RDP). Таким чином, ми можемо просто побачити повідомлення із запитом на підключення, але якщо ви декодуєте пакет як TLS, ви можете бачити відбитки клієнта TLSHello та JA3. Можна декодувати певний порт як інший протокол(див. Додаток Є).

2 Вектор розвитку створеного програмного продукту FATT:

- Підтримка нових протоколів (ETF QUIC, MySQL, MSSQL, SMTP, SMB).
- Реалізація модуля візуалізації(UI частини програмного продукту).

РОЗДІЛ 3. МОДЕЛЮВАННЯ МЕРЕЖЕВОГО ТРАФІКУ

3.1.Огляд генераторів мережевого трафіку

При аналізі мережевого трафіку можна можна діагностувати проблеми та вияляти вразливості працюючої системи. Після внесення відповідних змін, які закривають ці вразливості та вирішують проблеми. Постає питання відповідної перевірки правильності та повноти рішення, маючи відповідний дамп після аналізу, ми можемо це зробити. Але як уникнути відповідних проблем на етапі розробки? В цьому випадку нам на допомогу приходить техніка моделювання трафіку.

Існуючі генератори трафіку зосереджуються насамперед на підтримці статистичного розподілу різних характеристик мережевого трафіку, де можна розпізнати три основні підходи. Генератори трафіку на рівні пакетів, такі як iPerf [5], базуються на часі взаємодії і розмірі пакету. Розмір кожного відправленого пакету, а також час, що минає між наступними пакетами, вибираються користувачем, як правило, шляхом визначення статистичних розподілів для кожної змінної. Ці генератори критикуються за неточність [9]. Крім того, вони використовуються для перевірки працездатності та масштабованості мережеских інструментів, а трафіку, який вони генерують, бракує багатства та різноманітності пакетних потоків, що спостерігаються в реальних організаційних мережах; такі інструменти зосереджуються на тестах ефективності, а не на відтворенні характеру трафіку.

Генератори трафіку на рівні додатків імітують поведінку конкретних мережеских додатків щодо трафіку, який вони генерують. Наприклад, Surge [6] - це генератор завантаженості мережі, побудований за допомогою інтеграції розподілів функцій, витягнутих із використання цієї мережі. Ці інструменти, як правило, фокусуються на створенні послідовностей запитів на рівні додатків,

які призводять до мережевого трафіку з аналогічними статистичними даними трафіку, ніж реального трафіку в модельованому додатку. Хоча корисні для оцінки поведінки та продуктивності хост-систем, ці інструменти відтворюють лише один тип трафіку додатків, а не різноманітність трафіку, що переглядається в Інтернеті.

Генератори трафіку на рівні потоку, такі як Nagroop [7], відтворюють потоки трафіку (де потік визначається як серія пакетів між заданою парою IP/порт), що є представником тих, що переглядаються в Інтернет-трафіку, виходячи з восьми розподільних характеристик TCP та UDP потоки. Параметри цих дистрибутивів можуть бути автоматично вилучені з даних, зібраних з живого трафіку. Ці функції дозволяють Nagroop створювати статистичні навантаження, які не залежать від будь-якої конкретної програми. Тим не менш, статистика лише для восьми ресурсів зберігається в генерованому трафіку. Такі рамки, як Swing [8], намагаються подолати цю слабкість, враховуючи взаємодію між декількома шарами (користувачами, сесіями, з'єднаннями та мережевими характеристиками), щоб краще зберегти більше властивостей генерованого трафіку. Тим не менш, всі вищезазначені методи в основному зберігають статистику різних характеристик руху, але не мають можливості зберегти загальні послідовності в межах трафіку [18].

Дутта та ін. [4] представити основу для моделювання ботів користувачів, наслідуючи дії реальних користувачів, і продемонструвати його успішну реалізацію для виявлення вторгнень. Хоча їх робота базується на заздалегідь визначених правилах, ми підійдемо до аналогічного завдання з машинними методами навчання, які дозволять автоматизувати витяг реальної поведінки користувачів.

3.2. Формування ідеї

Використання генераторів мережевого трафіку в останні роки зростає, головним чином, через обмеження конфіденційності персональної інформації,

які обмежують використання реальних даних. Синтетичні дані можна вважати «фальшивими» даними, створеними з «реальних» даних. Перевага його впливає з його основи в реальних даних і реальних розподілах, які роблять його практично невідмінним від вихідних даних. Її імпульс, в умовах конфіденційності, впливає з того, що багато разів юридично неможливо ділитися реальними даними, але насправді анонімізовані дані є недостатньо корисними. У ці моменти встановлення синтетичного набору даних може представляти найкраще рішення обох світів - дані, якими можна ділитися, але все ж нагадують вихідні дані.

«Корисність» синтетичних даних була підтверджена такими дослідженнями, як [1–4]. У цьому дослідженні ми вивчаємо використання синтетичних даних для іншого типу даних, даних мережевого трафіку. Основна ідея запропонованого методу полягає в підготовці генеративної моделі, заснованої на даних обмеженої кількості користувачів, що погоджуються, а потім відповідно генерувати більш широкий і різноманітний трафік.

Створений трафік, який зберігає важливі особливості початкового трафіку, включаючи розподіл різних користувачів, додатків та мережевих властивостей, може бути корисним у багатьох напрямках. Наприклад, генерований трафік може слугувати вхідним фактором для аналізу вищого рівня, класифікації потоку та виявлення аномалії. Генератори мережевого трафіку також критично важливі в умовах тестування, де їх можна використовувати для проведення аналізу чи для оцінки поведінки та продуктивності нових систем та реального мережевого обладнання в корпоративних середовищах. Наприклад, інструмент вимірювання пропускної здатності може використовуватися для генерованих даних, що підтримують кількість пакетів. Ще одне використання генерованого трафіку в симульованих середовищах, де потрібен реалістичний фоновий трафік. Без можливості постійно створювати нові тестові умови (генеровані дані) в мережі, системи ризикують не впоратися з несподіваною поведінкою і погано працювати. Тому потрібні відповідні методи для створення масштабованого, регульованого та репрезентативного мережевого трафіку.

Існуючі рішення [5–8] були зосереджені на генеруванні статистично репрезентативного трафіку, але, наскільки нам відомо, не існує системи, яка б також зберігала послідовність моделей мережевої діяльності. Щоб краще зрозуміти значення збереження послідовностей мережевої активності, розглянемо наступний приклад: завантаження PDF-файлу з певного сайту зазвичай виконується після читання електронної пошти з папки "Вхідні" Gmail. Навіть якщо аналіз записаного трафіку покаже, що ця закономірність є статистично значущою, жоден із існуючих методів генерування штучного трафіку не збереже її. Тому система виявлення вторгнень (IDS), оцінена за допомогою генерованого трафіку, може не виявити можливе джерело атаки та виявити слабкішу ефективність, ніж це було б для вихідного трафіку.

Це призводить до основної мети дослідження в цьому розділі. Пропонується ідея Генератора мережевого трафіку, основу для збурення записаного мережевого трафіку з метою генерування різноманітного, але реалістичного фонових трафіку для сценаріїв моделювання мережі. NTG використовуватиме сотні функцій для підтримки якомога більшої кількості атрибутів вихідного трафіку, що забезпечить надійне рішення, яке не обмежується конкретним протоколом або додатком. Пакетні потоки кластеризовані для ідентифікації подібних мережевих дій, а потім методи машинного навчання використовуються для створення послідовних зразків мережевих дій, які будуть утримуватися в генерованому трафіку.

3.3.Прототип генератора мережевого трафіку

Прототип генератора мережевого трафіку включає чотири фази, що складаються з взаємопов'язаних наборів компонентів, як зображено на рисунку 3.1., попередня обробка мережевого трафіку, кластеризація потоків для подібних дій, моделювання послідовностей діяльності та генерація трафіку.

Фази генератора мережевого трафіку:

- вихідний трафік попередньо обробляється на агреговані потоки;

- потоки кластеризовані в групи мережевої діяльності;
- модель послідовності навчається для витягування загальних зразків на основі послідовностей мережевих дій (для цього завдання підтримується кілька методів);
- генеруються нові послідовності мережевих дій і потім перетворюються на вибірккові потоки пакетів, які відповідають кожній діяльності.

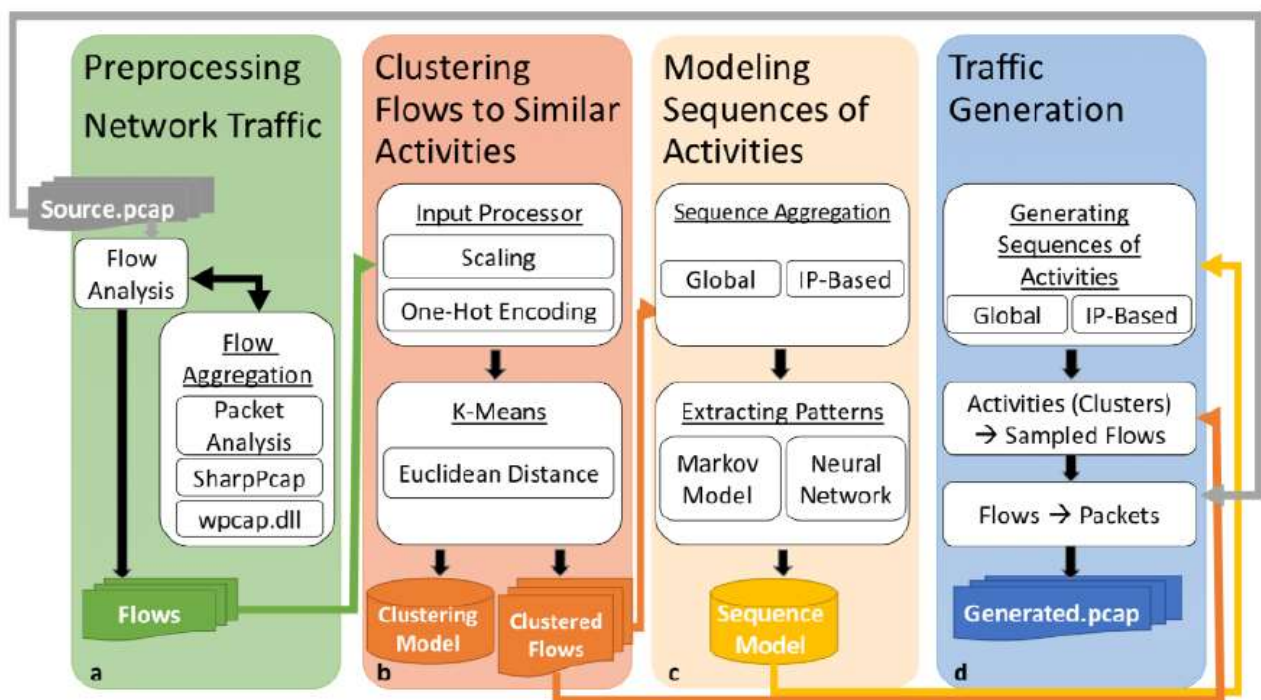


Рис. 3.1. Фази генератора мережевого трафіку

На етапі попередньої обробки ми зосередилися на двох найпоширеніших протоколах прикладного рівня (DNS та HTTP). Тим не менш, прототип легко розширити, щоб включити додаткові протоколи шляхом вилучення додаткових функцій. Після попередньої обробки записи потоків аналізуються далі для вилучення подібних дій. На наступній фазі послідовності діяльності моделюються за допомогою моделі Маркова або нейронних мереж, і, нарешті, на останній фазі тренувану модель послідовностей використовують для генерування штучного трафіку.

Основна сутність, що розглядається — потік трафіку; цей термін відноситься до послідовності пакетів, які були передані між двома вузлами

протягом одного сеансу, при цьому два вузли являють собою унікальний 4-пакет, що складається з вихідних та цільових IP-адрес та портів. Сеанс TCP починається з успішного handshake-у і закінчується таймаутом або пакетом із прапорцем RST або FIN від однорангового. Оскільки в цьому протоколі вузли посилають пакети по черзі, генеруються два потоки, по одному для кожного напрямку сеансу. Сеанс UDP складається з усіх пакетів, що надсилаються від клієнта до сервера до тих пір, поки не буде досягнуто визначеного часу простого зв'язку або максимальної граничної тривалості. Тому для кожного сеансу генерується один потік.

Визначили 205 унікальних функцій, які можуть формувати поведінку мережевого трафіку. Ми витягли функції з трьох рівнів:

- 1) рівень пакету;
- 2) рівень потоку;
- 3) рівень застосування.

Виділені функції детально описані в таблиці 1. Особливості рівня пакету витягуються з кожного пакету на рівні транспортного шару. Ми вилучили п'ятнадцять атрибутів на основі TCP та одну функцію на основі UDP. Кожна необроблена функція була перетворена в набір агрегативних ознак, які описують кожну особливість у пакетах заданого потоку. Наприклад, "розмір пакету" дістається до одинадцяти сукупних ознак, таких як середній розмір пакету (середня кількість розміру пакету в потоці), ентропія розміру пакета і т.д. Особливості рівня потоку, природно, витягуються з потоку. Ми витягли шість функцій рівня потоку; прикладом такої функції є кількість пакетів у потоці. Особливості рівня додатків – це функції, які відображаються на рівні додатків, вони стосуються лише певних програм. Наприклад, агент HTTP користувача буде вилучений для потоків, які підходять до HTTP-додатків. Ми витягли вісім функцій на основі DNS, вісім функцій на основі HTTP та сім функцій на основі SSL У таблиці 3.1 наведено кілька прикладів функцій, які відображаються в різних протоколах мережевого трафіку.

Таблиця 3.1

Функції на різних рівнях протоколів

Level	Features
Flow	time of day, packet interarrival time, the number of packets
TCP	time to live, seq num, ack num
UDP	checksum invalid
DNS	additional records, canon names, response count
HTTP	cookie, unique content types, bytes

Оскільки потік представляє певну мережеву діяльність від початку до кінця, ми можемо групувати подібні дії в кластери, які представляють типи мережевих дій. Цей крок необхідний для того, щоб дозволити виготовлення шаблонів послідовностей мережевих дій. Доступні різноманітні алгоритми кластеризації. Вибрано алгоритм k-means, оскільки це швидкий і простий алгоритм кластеризації. Для вимірювання відстані між потоками необхідно вибрати міру відстані. Ми приймаємо традиційну евклідову відстань, яка є однією з найбільш часто використовуваних метрик для кластерних проблем. Враховуючи набір потоків $F = \{f_1, \dots, f_n\}$, $f_i \in RM$, з характеристиками M на один потік, відстань між двома потоками становить:

$$dist(f_i, f_j) = \sqrt{\sum_{d=1}^{|M|} (f_i^d - f_j^d)^2} \quad (3.1)$$

Алгоритм k-means ділить набір потоків на K непересічних кластерів. Для кожного кластера алгоритм максимізує однорідність усередині кластера, мінімізуючи суму квадратичних помилок відстані між кожним потоком f_j та центром його кластера c_i :

$$E = \sum_{i=1}^K \sum_{f_j \in c_i} (dist(c_i, f_j))^2 \quad (3.2)$$

Алгоритм k-means починається з випадкової ініціалізації центрів кластерів і спрямовується до мінімальної похибки, де в кожній ітерації потоки присвоюються кластерам з найближчими центрами, після чого центри кластерів перераховуються. Зазвичай алгоритм конвергується в стабільні кластери в межах невеликої кількості ітерацій. Ми провели кілька додаткових кроків попередньої обробки, щоб переконатися, що набір потоків є відповідним завданням кластеризації; це включає перетворення номінальних ознак у числові ознаки методом одноразової бінаризації, а також масштабування ознак до одиничної дисперсії.

Потрібно розглянути різні варіанти агрегації моделей послідовності мереж. У зв'язку з цим ми зіткнулися з деякими питаннями. Чи має сенс підтримувати послідовність дій, що виконуються всіма членами мережі, чи слід зберігати послідовності дій, виконані парами комунікаційних IP-адрес (наприклад, послідовність дій від вихідного IP-адреси до певного сервера призначення)? Ми вивчаємо обидва ці параметри агрегації моделі послідовності мережеских дій. Перший (глобальний) відноситься до всього трафіку як до однієї послідовності, а другий (на основі IP) стосується кожної пари комунікаційних IP-адрес як окремої послідовності. У цьому дослідженні ми, серед іншого, вивчаємо, яка з двох агрегацій більше підходить для наших цілей.

Послідовність s - це упорядкований перелік мережеских дій, представлений ідентифікаторами кластерів, підключених до потоків, що відбувалися послідовно: $s = \{c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_i\}$, де $i \in K$. У глобальній агрегації всі потоки трафіку належать одній послідовності, тоді як в агрегації на основі IP потоки, що належать одній і тій же послідовності, поділяють джерело та цільові IP адреси. Час першого пакету в кожному потоці t_i диктує порядок дій у послідовності. Існує перехід між двома послідовними діями в межах однієї послідовності; час переходу між i -ою активністю і наступною активністю дорівнює $t_{i+1} - t_i$. Послідовні стани можуть бути однаковими (наприклад: $c_3 \rightarrow c_2 \rightarrow c_2$). Рисунок ілюструє різницю між агрегацією послідовностей на основі глобальної та IP-адреси. У глобальній агрегації всі потоки складаються з однієї

послідовності (жовта стрілка вказує їх порядок). У агрегації на основі IP створюється послідовність для кожної унікальної пари вихідних та цільових IP-адрес (рис. 3.2.). Оскільки в наборі прикладів є дві унікальні пари, утворюються дві послідовності: перша складається з трьох потоків, витягнутих з трафіку, які пройшли з 1.1.1.2 до 2.2.2.1 (позначені червоним кольором), а друга складається з двох потоків, які пройшли від 2.2.2.3 до 2.2.2.1 (позначено синім кольором). У агрегації на основі IP ми додаємо стартовий і кінцевий стани навколо кожної послідовності, щоб дати змогу вибирати початковий стан і передбачати кінець послідовності (вивчення розмірів послідовностей) [27].

Шукаючи сильний алгоритм вилучення моделі послідовностей, у цьому дослідженні ми вивчаємо два методи: модель Маркова(див. Додаток Ж) та модель нейронної мови (див. Додаток З).

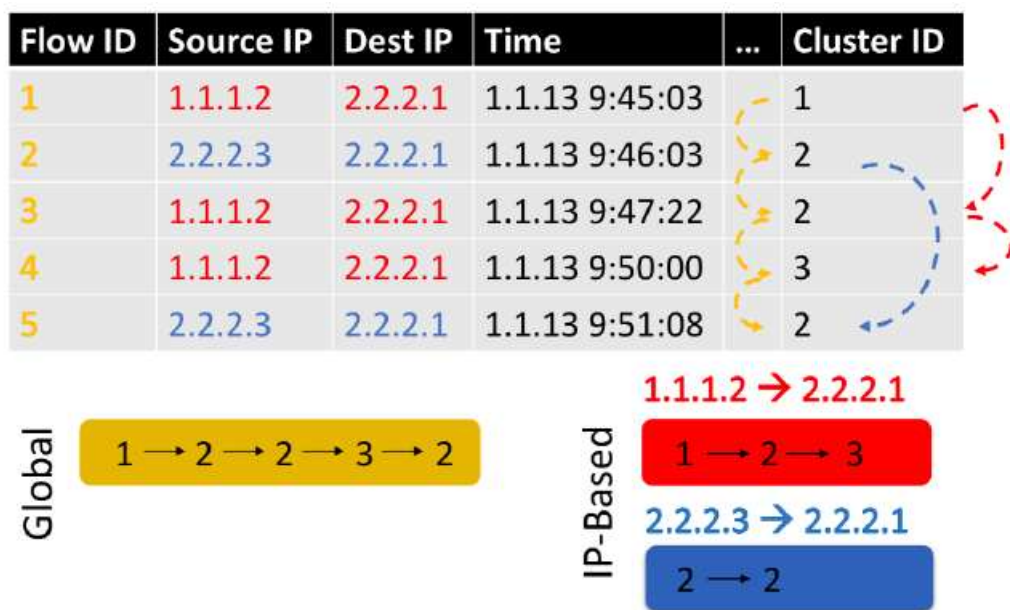


Рис. 3.2. Дві послідовності агрегації: на основі глобальної та IP-адреси.

У глобальній агрегації (жовтий) всі потоки в трафіку належать одній і тій же послідовності, тоді як в агрегації на основі IP (червоний і синій) створюється послідовність для кожної унікальної пари IP-адрес джерела та призначення. Послідовність складається з ідентифікаторів кластера, приєднаних до відповідних потоків і впорядкована часом першого пакету в кожному потоці.

Ми пропонуємо генеративний метод, відповідний навченим моделям. По суті, нова послідовність мережових дій (представлена ідентифікаторами кластера) створюється шляхом "склеювання" дуже коротких та фрагментів мережових дій, які перекриваються, що часто зустрічаються у навчальних даних [24]. Правила отримання ймовірності наступної мережової активності залежать від використовуваного методу (модель Маркова або модель нейронної мови) і є дещо неявними.

З метою генерації мережового трафіку створюються послідовності дій (кластери); потім потоки, відповідні кожному кластеру, відбираються з вихідного трафіку, і в кінцевому підсумку пакети цих потоків передаються в порядку, зазначеному генерованими послідовностями. У глобальній агрегації послідовностей модель послідовності навчається на потоках всіх членів мережі, і відповідно генерується єдина глобальна послідовність, яка імітує мережові дії всіх однорангових мереж, упорядкованих за часом їх виникнення. Послідовність завершується, коли досягнутий приблизний час закінчення; отже, на розмір послідовності в основному впливають розрахункові часи передачі, які базуються на статистиці часу переходу, яка була вилучена з вихідного трафіку. У агрегації на основі IP створюється послідовність для кожної пари IP-адрес (адреси регенеруються), де послідовності закінчуються, коли доходить до стану кінця. Використовуючи модель нейронної мови, контекст m попередніх кластерів повинен бути оновлений у кожній ітерації для прогнозування наступного кластера; при використанні моделі Маркова попередній кластер оновлюється таким же чином. Алгоритм генерації трафіку викладений в формальному описі нижче (рис. 3.3).

```

1  $D^* \leftarrow \text{ExtractFlows}(D)$ ;
2  $D^* \leftarrow \text{AttachClusters}(D^*, M_{\text{Clusters}})$ ;
3  $t_{\text{start}}, t_{\text{end}} \leftarrow \text{FirstTime}(D), \text{LastTime}(D)$ ;
4  $\text{trans} \leftarrow \text{ExtractHist}(\text{TransitionTimes}(D^*), \text{bins} = \sqrt{|D^*|})$ ; // transition times between flows
5 if using global aggregation then
6    $c \leftarrow \text{SampleFirstCluster}(M_{\text{Sequences}})$ ;
7    $t \leftarrow t_{\text{start}}$ ;
8   while  $t < t_{\text{end}}$  do
9      $f \leftarrow \text{SampleFlow}(D^*, c)$ ;
10     $t \leftarrow t + \text{Sample}(\text{trans})$ ;
11     $\text{context} \leftarrow \text{Update}(\text{context}, c)$ ;
12     $c \leftarrow \text{PredictCluster}(M_{\text{Sequences}}, \text{context})$ ;
13    Add  $\text{GetPackets}(f, D)$  to results set;
14 else
15   // If using IP-based aggregation
16   for each pair  $IP_{\text{src}}, IP_{\text{dest}}$  in  $D$  do
17      $\text{context} \leftarrow \text{Update}(\text{context}, \text{start})$ ;
18      $c \leftarrow \text{PredictCluster}(M_{\text{Sequences}}, \text{context})$ ;
19      $IP_{\text{src}}^*, IP_{\text{dest}}^* \leftarrow \text{GenerateIPs}()$ ;
20      $t \leftarrow t_{\text{start}}$ ;
21     while  $c \neq \text{end}$  do
22        $f \leftarrow \text{SampleFlow}(D^*, c)$ ;
23        $t \leftarrow t + \text{Sample}(\text{trans})$ ;
24       Add  $\text{SetIPs}(\text{GetPackets}(f, D), IP_{\text{src}}^*, IP_{\text{dest}}^*)$  to results set;
25        $\text{context} \leftarrow \text{Update}(\text{context}, c)$ ;
26        $c \leftarrow \text{PredictCluster}(M_{\text{Sequences}}, \text{context})$ ;

```

Рис. 3.3. Алгоритм генерації мережевого трафіку

Його вхід містить дані мережевого трафіку (D), які використовуються для реконструкції записаних пакетів при генерації нового трафіку, а також модель кластеризації (M_{Clusters}) та модель послідовностей ($M_{\text{Sequences}}$).

3.4. Методи оцінки генератора трафіку

Нижче буде використано оцінку запропонованої нами основи та порівняння досліджених методів. Ми використовуємо бал Silhouette, щоб оцінити, наскільки добре ми кластеризували потоки.

Багато функцій було витягнуто з мережевого трафіку, і щоб продемонструвати, наскільки добре збережений метод кожного з них, ми використовуємо тести гіпотез, щоб перевірити, чи джерело та синтезовані дані кожної функції походять з одного розподілу [25]. Ми також використовували n-гранні невдачі, щоб перевірити, чи збережено розподіл послідовності значень ознак.

Оцінка силуету вимірює, наскільки член схожий на його кластер (згрупованість) порівняно з іншими кластерами (розділення).

Оцінка коливається від -1 до $+1$, при цьому більш високі бали вказують на те, що член добре згрупований (подібно до самого кластера та далеко від сусідніх кластерів). Високі бали для більшості членів кластеру вказують на те, що конфігурація кластеру є відповідною. Оцінка силуету для одного потоку f_i розраховується так:

$$s(i) = \frac{b(f_i) - a(f_i)}{\max(a(f_i), b(f_i))} \quad (3.3)$$

де $a(f_i)$ - середня відстань від f_i до всіх інших потоків всередині його кластеру, а $b(f_i)$ - найменша середня відстань f_i до всіх потоків у будь-якому іншому кластері (відстань до другого найближчого кластера). Ми використовували середній бал силуету кластерних потоків для оцінки структури конфігурації кластеру та налаштування кількості кластерів.

Тест гіпотези К.С. (Колмогорова-Смірнова) — непараметричний тест рівності безперервних одновимірних розподілів ймовірностей, який можна використовувати для порівняння двох вибірок. Він кількісно визначає відстань між емпіричними функціями розподілу двох зразків. Нульовий розподіл цієї статистики обчислюється під нульовою гіпотезою про те, що вибірки беруться з одного розподілу. Ми використовуємо цей тест, щоб перевірити, чи неперервна функція має однаковий розподіл як у вихідному, так і у генерованому трафіку.

Тести гіпотез K-mean Андерсона-Дарлінга AD(Anderson-Darling) використовуються для визначення того, чи можна оцінити кілька колекцій спостережень як таких, що надходять із однієї сукупності, де функцію розподілу не потрібно вказувати. Це модифікація тесту KS. Ми використовуємо цей тест, щоб перевірити, чи має номінальна ознака однаковий розподіл як у вихідному, так і у генерованому трафіку.

Заплутаність вимірює успіх розподілу ймовірностей або моделі ймовірності передбачення вибірки. Він може бути використаний для порівняння ймовірнісних моделей. Низька заплутаність вказує на те, що

розподіл ймовірностей добре прогнозує вибірку. При обробці природною мовою заплутаність часто використовується для оцінки мовних моделей, порівнюючи n-гранні невдачі в оригінальних та створених даних. Цю ідею адаптуємо для вивчення збереження послідовностей мережових дій для кожного IP-джерела. Заплутаність дискретної випадкової величини X визначається як:

$$2^{H(X)} = 2^{-\sum_x p(x) \log_2(p(x))} \quad (3.4)$$

де $H(p)$ - ентропія (у бітах) розподілу по кожному можливому значенню x .

3.5.Оцінка генератора трафіку

Дані розрахунки проводилися на двох уявних колекціях мережевого трафіку. Різні характеристики цих двох колекцій наведені в таблиці 3.5. Як видно з таблиці, дві колекції трафіку мають різні характеристики; у них різний час збору, і в трафіку EMC майже втричі більше потоків, але потоки в трафіку EMC набагато коротші, ніж потоки BGU, і містять набагато менше пакетів [23]. Крім того, хоча трафік EMC містить приблизно половину кількості послідовності порівняно з трафіком BGU, він містить у шість разів більше комунікаційних пар IP, що ускладнює трафік, принаймні для агрегації на основі IP.

Таблиця 3.2.

Різні характеристики трафіку, використовуваного для експериментів

	BGU	EMC
Розмір	20.1 Гб	2.79 Гб
Кількість пакетів	193 50 505	8 700 370
Кількість потоків	190 462	549 350
Source IPs	1 329	633
Destination IPs	1 309	435
Source та Destination	3 322	24 760
Мінімальний час	1.10.2022 1:05:57	13.06.2022 07:38:27
Максимальний час	2.10.2022 1:11:07	13.06.2022 12:59:14
Розмір послідовності	57 721 потоків	22 367 потоків

У наборі наших оцінок зосередимося на оцінці та адаптації моделі кластеризації. Ми розглянули три різні конфігурації кластеризації, змінивши кількість кластерів для кожної конфігурації ($K = 100, 500, 1000$). Попередньо оброблений трафік BGU був кластеризований відповідно до кожної з цих конфігурацій, використовуючи k-means реалізацію scikit-learn. Для кожної оцінюваної конфігурації послідовну модель потім навчали за допомогою глобальної моделі Маркова і можливо використовуватиметься для генерації зразків трафіку.

Ми оцінювали якість структури кластерів за допомогою оцінки силуету, як описано на рис. 3.4. Нагадаємо, що більш високі показники вказують на чіткіший поділ на кластери, тому результати показують, що для досліджуваних конфігурацій збільшення кількості кластерів (K) покращується структура кластерів, однак це відбувається за ціною істотно збільшуваних модельних часів навчання, як видно на рис. 3.5.

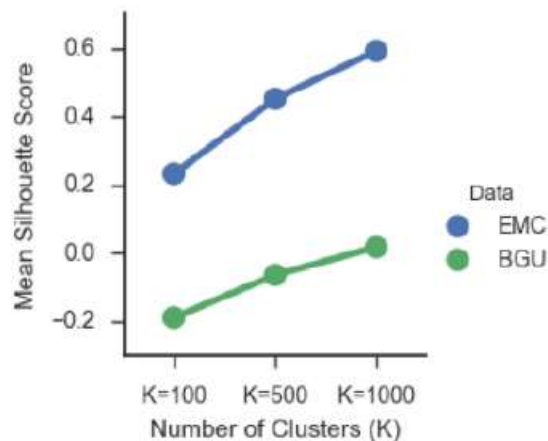


Рис. 3.4. Середні показники силуету проти кількості кластерів.

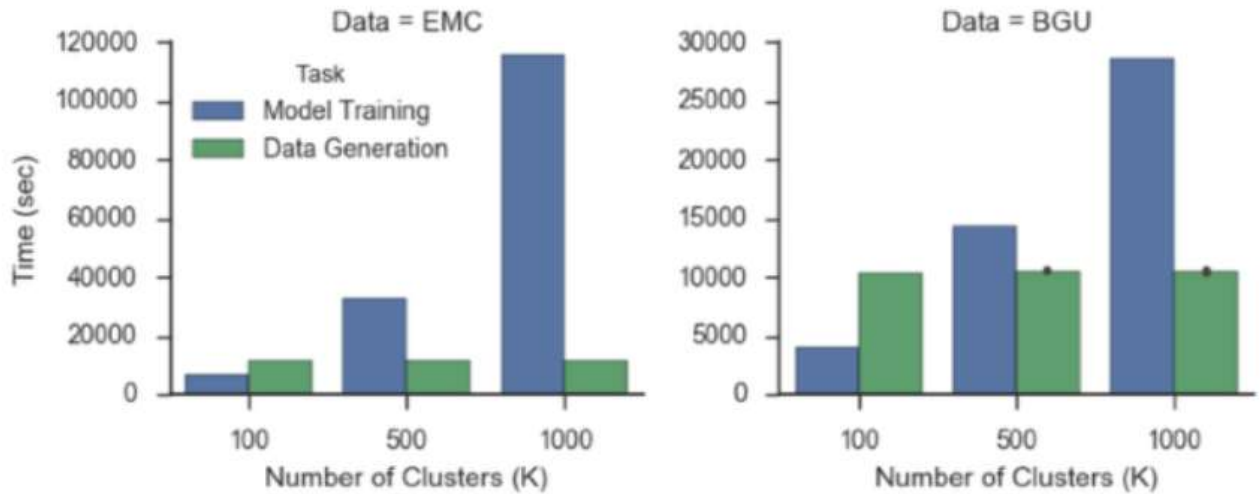


Рис. 3.5. Час виконання порівняно з кількістю кластерів для модельних завдань (синій) та формування даних (зелений).

Далі ми мали на меті зрозуміти, яка з особливостей найбільше вплинула на процес кластеризації. Для цього завдання ми використовували класифікатор випадкових дерев, який зазвичай використовується для подібних завдань. Для кожної досліджуваної конфігурації кластеризації ми підготували модель Random Forest для класифікації попередньо обробленого трафіку на кластери, де цільовою змінною (clusterID) є ідентифікатор кластеру, отриманий у фазі кластеризації. Потім ми витягли топ-10 найвпливовіших особливостей у навчній моделі Random Forest (вимірюється на основі інформаційного посилення різних функцій). У таблиці на рисунку 3.6 наведено інформацію про топ-10 найвпливовіших функцій для кожної дослідженої конфігурації на основі трафіку BGU. Перші шість особливостей у всіх конфігураціях базуються на часі дня, коли відбувся потік (функція денного часу). Функція TCP SeqID також є домінуючою для 500 та 1000 кластерів, тоді як для 100 кластерів переважна функція розміру вікна TCP [28]. День тижня, в якому відбувся потік (особливість тижня, день), також впливає при використанні 100 або 500 кластерів. Досить схожі результати були отримані для набору даних EMC.

Rank	K=100	K=500	K=1000
1	day time last	day time min	day time min
2	day time min	day time sum	day time mean
3	day time sum	day time last	day time first
4	day time first	day time first	day time sum
5	day time max	day time max	day time max
6	day time mean	day time mean	day time last
7	TCP win size sum	IP ttl sum	TCP seqID first
8	week day first	TCP seqID mean	TCP seqID sum
9	TCP win size thirdQ	week day sum	TCP seqID last
10	packet size mean	TCP seqID last	TCP seqID mean

Рис. 3.6. Топ-10 найвпливовіших функцій для кластеризації трафіку BGU з використанням різної кількості кластерів (K). (Особливості, витягнуті з одного і того ж атрибута, відображаються в одному кольорі.)

Інший спосіб оцінити якість результатів кластеризації - перевірити, наскільки збережені ймовірності різних ознак. Для кожної з розглянутих конфігурацій кластеризації та для кожної з 197 числових ознак, що відображаються в даних, ми використовували двопробний тест KS, щоб перевірити, чи розподіл функції у вихідному та створеному трафіку був однаковим на рівні значущості 0,01. Аналогічним чином ми використовували тест AD на двох зразках, щоб перевірити, чи збережено розподіл семи номінальних ознак. На рисунку 3.6 показані результати, отримані для кожної тестованої конфігурації кластера. Сто кластерів працювали найкраще, зберігаючи розподіл більшості функцій. Використання більше 100 кластерів послаблює результати, ймовірно, тому, що рідкісні кластери можуть не відображатися в генерованому трафіку, так що відсутність можливих значень для членів цих кластерів призводить до зміни статистики, але навіть для 1000 кластерів, щонайменше, 50% рис зберігається.

Для того, щоб оцінити, наскільки добре збережені шаблони послідовностей у генерованому трафіку, ми вимірювали невдачі для n-грамів з трьома різними розмірами ($n = 2, 3, 4$) та обчислювали різниці між неприємностями у вихідному та генерованому трафіку. Взагалі важче зберегти

послідовності для довгих підрядів, тому заплутаність та їх відмінності зазвичай збільшуються з n . Як показано на Рисунку 3.7, 100 кластерів призводять до найменшої різниці між n -гранними невдачами у вихідних та згенерованих даних, і тому ця кількість кластерів є найкращою конфігурацією кластеризації для збереження шаблонів послідовностей. Це розумно, оскільки зменшення кількості параметрів полегшує збереження розподілу [26]. Використання 100 кластерів призводить до найкращих показників як з точки зору якості генерованого трафіку, так і часу роботи.

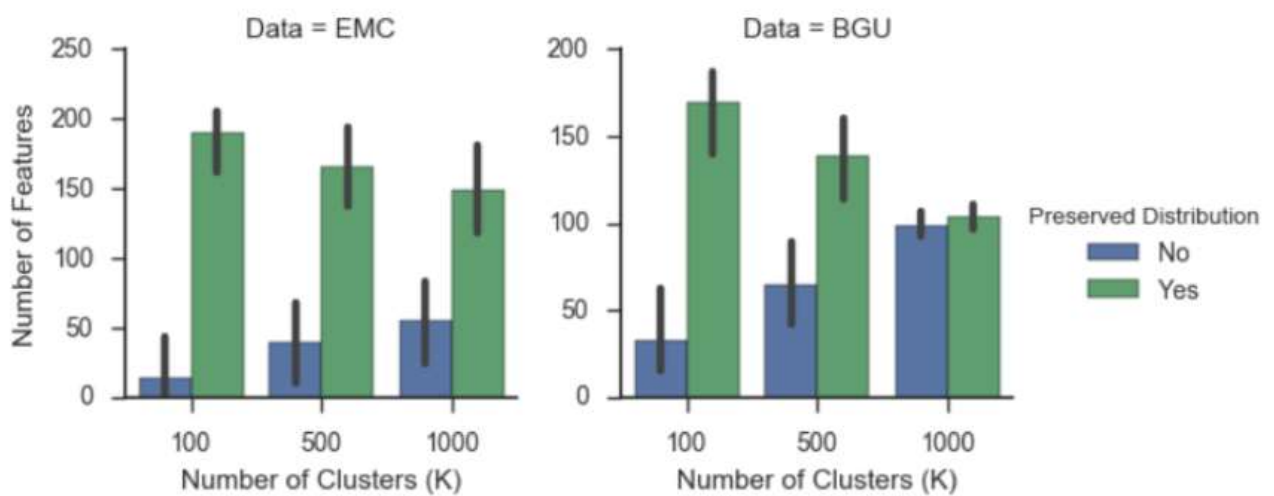


Рис. 3.7. Кількість функцій, розподіл яких однаковий (зелений) або різний (синій) як у вихідному, так і в генерованому трафіку, згідно з тестами KS та AD ($\alpha = 0,01$) проти кількості кластерів.

ВИСНОВКИ

Таким чином формуємо результати даної роботи:

1. Досліджено процес аналізу трафіку.

2. На основі цього дослідження проведено порівняльний аналіз існуючих засобів аналізу трафіку. Реалізовано базовий функціонал пакетного аналізатора для деяких мережевих протоколів з унікальною реалізацією методів створення відбитків.

3. З розробленого аналізатора з'являється потреба моделювання мережевого трафіку відповідно проведено дослідження моделювання трафіку. Базуючись на цьому розроблена та досліджена модель моделювання мережевого трафіку зі збереженням послідовності. Переваги розробленого пакетного аналізатора, які виходять з використання відповідних методів створення відбитків:

- Можна виявити, контролювати і досліджувати спроби взлому з більш високим рівнем деталізації, ніж джерело IP.

- Виявити приховану ексфільтрацію даних у компонентах набору клієнтських алгоритмів. У цьому випадку спеціально кодований SSH- клієнт може надсилати вихідні дані з надійного середовища до менш надійного середовища в межах серії пакетів SSH_MSG_KEXINIT. У сценарії, подібному до більш відомої ексфільтрації через DNS, дані можуть бути надіслані у вигляді серії спроб, але неповних і незамкнених підключень до SSH-сервера. Такі спроби не реєструються навіть відомими аналізаторами пакетів або в системах кінцевих точок. Виявлення цього стилю ексфільтрації тепер може бути легко здійснено за допомогою виявлення аномалії або оповіщення для клієнтів SSH з декількома різними помилками.

- Створення додаткового рівня управління клієнтськими додатками, наприклад, ви можете заблокувати підключення всіх клієнтів до SSH-сервера, який знаходиться поза затвердженим відомим набором значень `hassh`.

- Виявлення пристроїв, які мають хеш-файли, які, як відомо, належать до вбудованих систем ІОТ. Приклади можуть включати камери, мікрофони, кейлоггери, які можна легко приховати від перегляду та спокійного спілкування по зашифрованим каналам.

По завершенню відповідного дослідження планується до вектору розвитку FATТ додати реалізацію моделі моделювання мережевого трафіку зі збереженням послідовності. Таким чином розширивши можливості новоствореного продукту не лише аналізом трафіку, а відповідної можливості його генерації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1.Пакетні сніфери. URL: <https://www.inter-nauka.com/uploads/public/15895669733068.pdf> (дата звернення 01.06.2023)
- 2.Mike Cloppert. An Overview Of Protocol Reverse-Engineering. URL: <https://digitalforensics.sans.org/blog/2012/07/03/an-overview-of-protocol-reverseengineering> (дата звернення 02.03.2023)
3. Що таке сніфер? URL: <https://www.avg.com/en/signal/what-is-sniffer> (дата звернення: 13.05.2023)
4. Stephen Northcat, Judy Nowak, Network Security Discovery. 3rd ed. New York: Sams Publishing, 2003. 356 p.
5. Кращі 10 сніферних пакетів та їх інструментів у 2020 URL: <https://www.dnsstuff.com/packet-sniffers> (дата звернення: 13.05.2023)
6. Orebaugh Angela ,Wireshark network protocol analyzer, 2006. 450 p.
- 7.The Bro Network Security Monitor. URL: <http://www.bro.org/> (дата звернення 07.03 .2023)
8. Wireshark. URL: <http://www.wireshark.org/> (дата звернення 17.04 . 2020)
9. Tcpdump. URL: <http://www.tcpdump.org/> (дата звернення 25.03 . 2020)
10. Wireshark Display Filter Reference. URL: <https://www.wireshark.org/docs/dfref/>, (дата звернення 21.04 .2023)
11. Wei K., Yang J., Fu Y. Single image reflection removal exploiting misaligned training data and network enhancements 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019. P. 8170–8179.
12. А. Ализар. Доля зашифрованого трафіка в інтернеті выросла в несколько раз. URL: <https://хакер.ru/2014/05/15/62500/> (дата звернення 05.04 .2023)
13. Сетевые модели OSI и TCP/IP. URL: http://www.quizful.net/post/osi_tcpip_layers (дата звернення 07.05 .2023)

14. Colasoft Packet Player. URL: http://www.colasoft.com/packet_player/ (дата звернення 07.05 .2023)
15. Du C., Kang B., Xu Z. Accurate and efficient video de-fencing using convolutional neural networks and temporal information. 2018 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2018. P. 1–6.
16. SSL/TLS. URL: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html#SECTION0032140000000000000000>, (дата звернення 12.04 .2023)
17. Microsoft SMB Protocol and CIFS Protocol Overview, URL: [https://msdn.microsoft.com/en-us/library/aa365233\(VS.85\).aspx](https://msdn.microsoft.com/en-us/library/aa365233(VS.85).aspx), (дата звернення 10.05 .2020)
18. Мэйволд Э. «Межсетевые экраны» // Безопасность сетей: Информация, ИНТУИТ, 2006. URL: <http://www.intuit.ru/studies/courses/102/102/lecture/2989>, (дата звернення 22.05 .2023)
19. Christopher Parsons. Deep Packet Inspection in Perspective: Tracing its lineage and surveillance potentials. URL: https://www.academia.edu/9603296/Deep_Packet_Inspection_in_Perspective_Tracing_its_lineage_and_surveillance_potentials , (дата звернення 30.05 .2023)
20. Documentation to DCE/RPC, URL: <http://www.dcerpc.org/documentation/>, (дата звернення 30.05 .2023)
21. Oink: a Collaboration of C/C++ Tools for Static Analysis and Source-to-Source Transformation. URL: <http://daniel-wilkerson.appspot.com/oink/index.html/>, (дата звернення 01.06 .2023)
22. Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). Montreal, QC, Canada, 17-19 Oct. 2016, IEEE. 410p.
23. Paul Barford, Mark Crovella, Paul Barford, and Mark Crovella. Generating representative Web workloads for network and server performance evaluation. New York, USA, 1998. 160p.

24. Alessio Botta, Alberto Dainotti, and Antonio Pescape. Do you trust your software-based traffic generator? Montreal, QC, Canada. IEEE Communications Magazine, 48(9): sep 2010. 165p.
25. Joel Sommers and Paul Barford. Self-configuring network traffic generation. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement . New York, USA, 2004.108p.
26. J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. New York: Wiley, 1979. 108p.
27. Xiaoming Zhou and Piet Van Mieghem. Reordering of IP Packets in Internet. 5th ed. France: PAM, 2004, 246p.
28. Arjuna Sathiaselan and Tomasz Radzik. Improving the performance of TCP in the case of packet reordering. France: IEEE. 2004. 73p

