ISSN 2707-1898 (print)
Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2022.02.061
Article received 05.10.2022 p.
Article accepted 17.10.2022 p.
UDK 004.715

Correspondence author
I. G. Tsmots
ivan.h.tsmots@lpnu.ua

І. Г. Цмоць, С. В. Теслюк
Національний університет "Львівська політехніка", м. Львів, Україна

МОДЕЛІ ТА ЗАСОБИ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДОСЛІДЖЕННЯ ТРАФІКУ КОМП'ЮТЕРНИХ МЕРЕЖ З ВИКОРИСТАННЯМ ФІЛЬТРА ПАКЕТІВ БЕРКЛІ

В роботі запропоновано використати інструмент Linux – фільтр пакетів Берклі (англ. *Berkeley Packet Filter*) для автоматизації дослідження трафіку в комп'ютерних мережах. Розроблено структуру програмного засобу, яка базується на модульному принципі, що дає змогу швидко вдосконалювати та модернізувати систему. Побудовано основні алгоритми функціонування програмного засобу, а саме: алгоритм опрацювання мережевого пакету з використанням фільтра пакетів Берклі та алгоритму функціонування програмного засобу із користувацького простору для завантаження Berkeley Packet Filter та налаштування комунікації із нею. Розроблено моделі дослідження динаміки функціонування програмного засобу, яка базується на теорії мереж Петрі. Внаслідок застосування моделей в процесі розроблення програмного засобу на підставі мереж Петрі – система працює коректно, а усі стани досяжні, тупики відсутні. Побудована імітаційна модель застосування інструмента Berkeley Packet Filter для автоматизації дослідження трафіку комп'ютерних мереж та розроблено скрипт для тестування розробленого програмного засобу. Розроблено програмне забезпечення системи моніторингу трафіку комп'ютерної мережі з використанням фільтра пакетів Берклі, яке використовує мови C, C++ та Python, що забезпечує збір, збереження опрацювання даних трафіку комп'ютерної мережі та подання результатів дослідження у формі зручній для користувача. Наведено результати тестування трафіку в комп'ютерних мережах в різних режимах нормального функціонування та при DDoS-атаках. Зокрема, приклад вихідних даних на боковій панелі із статистикою мережевого трафіку за тривалий період часу, приклад результатів з параметрами різкого скачка мережевого трафіку та приклад попереджувального повідомлення, який видасть програмний засіб на боковій панелі.

Ключові слова: алгоритм опрацювання мережевого пакету; алгоритму функціонування програмного засобу; автоматизація дослідження трафіку; мережа Петрі; імітаційна модель; DDoS-атака.

Вступ / Introduction

Щоденно безліч сервісів піддається атакам зловмисників, наслідки яких можуть тимчасово припинити роботу сервісу, або навіть завершитися потраплянням конфіденційних даних у чужі руки [1], [11]. Для запобігання таких неприємних ситуацій першим кроком є детальний моніторинг та аналіз мережевого трафіку, що постійно надходить. Відповідно, розроблення технологій, методів, моделей та програмних засобів для підвищення рівня захисту комп'ютерних мереж є актуальною завданням сьогодення.

Об'єкт дослідження – процес збирання, збереження та опрацювання даних трафіку комп'ютерної мережі.

Предмет дослідження – моделі та програмні засоби для збирання, збереження та опрацювання даних трафіку комп'ютерної мережі з використанням фільтра пакетів Берклі.

Мета роботи – підвищення ефективності захисту процесу передачі даних в інтернет мережі за допомогою унікального інструмента Linux – фільтра пакетів Берклі [10].

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- провести огляд літературних джерел за тематикою дослідження;
- розробити імітаційну модель та моделі дослідження динаміки автоматизованої системи дослідження трафіку комп'ютерних мереж з використанням фільтра пакетів Берклі;
- розробити програмний засіб, який використовує фільтр пакетів Берклі та забезпечує автоматизацію збирання, збереження та опрацювання даних про вхідний трафік та відображає загальну статистику.

Структура статті містить 5 розділів. В першому розділі проведено аналіз літературних джерел за тематикою дослідження. Другий розділ містить інформацію про розроблену структуру та основні алгоритми побудованого програмного засобу. Третій розділ містить дані про розроблене математичне забезпечення, а четвер-

тий розділ присвячений розробленню програмного забезпечення системи. В п'ятому розділі наведено результати дослідження.

Аналіз останніх досліджень та публікацій. Інструмент BPF типу XDP був вперше представлений у 1992 році [2], як зручний спосіб "вловлювання" мережевих пакетів із простору користувача. Він мав ряд переваг і було заявлено, що тогочасний BPF працює в 20 разів швидше, ніж інші способи опрацювання пакетів з user-space в ті часи.

Основні моменти, які зробили BPF ефективним способом внесення змін до поведінки ядра були: BPF працює як окрема віртуальна машина, оптимізована для роботи з центральними процесорами на підставі регістрів; інструмент BPF використовує аплікаційні буфери, що дає можливість уникнути копіювання інформації мережевих пакетів, яка необхідна для опрацювання; перед завантаженням скрипта із використаним інструментом BPF у простір ядра, верифікатор виконує різні перевірки, щоб затвердити, що скрипт безпечний для виконання та не потрапить у нескінченний цикл.

Протягом останніх років ентузіасти Linux розширили функціональність BPF [3], [14] до розширеної версії (також відома як eBPF, проте для скорочення її продовжують називати BPF, а попередню версію – як classical BPF (cBPF)). Нова версія BPF працює з 64-бітними регістрами, замість 32-бітних. Різноманітність подій, на які можна прикріпити виконання BPF теж розширилась і не обмежується тільки подіями пов'язаними із опрацюванням мережевих пакетів (наприклад, тепер можна почати виконання скрипта на підключення нового пристрою через USB).

Наразі, розмір завантаженої програми із використаним інструментом BPF обмежений 4 Кб. Якщо користувач хоче вийти за задані межі – можна реалізувати шляхом створення впорядкованих ланцюжків завантажених BPF скриптів одного типу [3]. Повідомлення у програмі з BPF можна створити за допомогою bpf_trace_printk [2], які будуть доступні серед повідомлень ядра.

Проведемо аналіз наявних інструментів із консольним інтерфейсом для збирання, збереження та опрацювання даних комп'ютерної мережі [17], оскільки більшість серверів розташовані віддалено і до них доступаються за допомогою SSH [13] через консоль. Різноманітність інструментів для Linux величезна, тому розглянемо найпопулярніші або ті, які пропонують більше унікальних функцій. Деякі, серед розглянутих інструментів, також використовують BPF.

Отже, перший інструмент – tcpdump [18] є найпопулярнішим консольним інструментом для аналізу мережевого трафіку, який забезпечує відфільтрування аналізу мережевого трафіку за IP адресою, інтерфейсом або портом, який може відобразити внутрішній зміст мережевого пакету, підтримує різні режими перегляду інформації та використовує libpcap [18] бібліотеку, яка своєю чергою використовує BPF.

Наступний розглянутий засіб є Nload [15], який підтримує моніторинг доступний для вхідного та вихідного мережевого трафіку окремо, показує загальну статистику таку як: поточне мережеве навантаження (Мбіти/секунду); середнє мережеве навантаження; мінімальне мережеве навантаження; максимальне мережеве навантаження та загальне мережеве навантаження.

Наступним програмним засобом, який варто взяти до уваги є Ifload / Iptraf [7]. Цей засіб дає змогу виміряти мережеве навантаження через окремі з'єднання сокетів і генерує загальну статистику.

Netlog [12] показує мережеве навантаження для визначеного процесу та сортує процеси за інтенсивністю мережевого навантаження. Здебільшого, програмні засоби із графічним інтерфейсом уже орієнтовані для моніторингу трафіку у групі серверів, є загалом більш функціональними і також часто комерційними.

Корисна інформація, яка відображається у деяких таких програмних засобах, розглянута в роботі [22] є: статус пристроїв у контрольованій мережі; візуальне зображення топології внутрішньої мережі; навантаження на процесор; використання пам'яті; активність вузлів; кількість з'єднань на кожен віртуальний сервер; моніторинг правил Firewall та загальна статистика.

Отже, наведений аналіз дає змогу стверджувати, що існує потреба в розробленні автоматизованої системи дослідження трафіку комп'ютерних мереж з використанням BPF, який також має відкритий код.

Результати дослідження та їх обговорення / Research results and their discussion

Розроблення структури та основних алгоритмів автоматизованої розробки та дослідження комп'ютерних мереж з використанням BPF. Розроблена структура програмного засобу зображена на рис. 1. Структура програмного засобу містить дві основні частини: машину сервісу та машину користувача. Машина сервісу містить програмні модулі, а саме: модуль опрацювання вхідних мережевих пакетів; мапи; читальні мапи; модуль агрегації даних і їх аналізу і сервер відправки даних. Машина користувача – частина програмного засобу, яка містить клієнтський модуль візуалізації даних у форматі зручному для користувача.

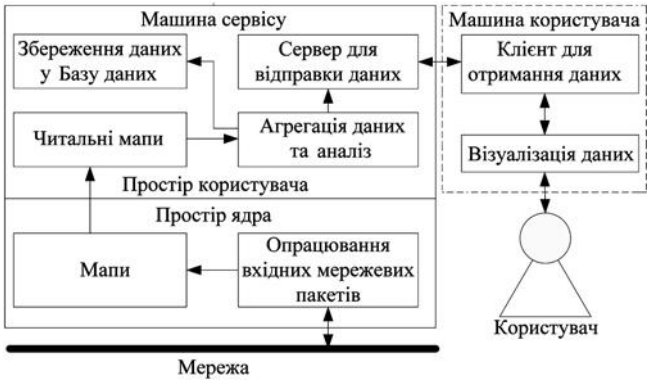

Рис. 1. Структурна схема програмного засобу / Structural diagram of the software tool

Модуль опрацювання вхідних мережевих пакетів – власне і є BPF програма (типу XDP, оскільки ми "виловлюємо" сирі мережеві пакети), яка прикріплена до визначеного мережевого інтерфейсу. Її роль полягає у тому, що при отриманні нового мережевого пакету, пакет розпаковується, пізніше необхідно прочитати визначені загальні дані (розміри, протокол і т.д.) та записати ці дані у мапу для передачі на подальше опрацювання. Ця частина програмного засобу виконується у просторі ядра.

Визначені мапи виконують роль сховища, для передачі даних між BPF програми (вони мають прямий дос-

туп) та частиною програмного засобу із користувацького простору (які доступаються до даних збережених у мапі через віддзеркалення мапи у користувацькому просторі – читальну мапу).

Віддзеркалення мап (читальні мапи), створені щоб програмні модулі із користувацького простору мали доступ до даних, переданих від BPF скрипта.

Частина програмного засобу створена для агрегації даних і аналіз виконується у просторі користувача. Також відповідає за завантаження скомпільованої BPF програми у простір ядра, встановлення комунікації із BPF програмою (через налаштування мап), і власне за зчитування отриманої інформації, її агрегування та виведення на консоль загальної статистики в реальному режимі часу.

Сервер для відправки даних дає можливість встановлювати сесію із клієнтами, які під'єднуються та надсилати агреговані дані в реальному режимі часу.

Машина користувача містить:

- Клієнтський модуль – призначений для отримання даних, який під'єднується до раніше вказаного сервера, встановлює сесію і, при надходженні нових даних, оновлює візуалізацію, враховуючи новоприбулі дані.
- Модуль візуалізації даних – призначений для відображення вхідних нових даних у вигляді графіків значень загальної статистики. Ці графіки оновлюються щоразу, після опрацювання нових даних, тобто відображають інформацію в режимі реального часу.

Розглянемо роботу системи із найнижчого рівня, тобто із BPF програми, яка власне опрацьовує кожен вхідний мережевий пакет визначеного інтерфейсу. Розпакування мережевого пакету із перевірками на вміст подальших даних має відбуватися із врахуванням будови Ethernet пакету. У блоці даних IP пакету також може знаходитись UDP, ICMP або інший пакет. Перевірка на протокол цієї частини пакету також здійснюють. Також варто зауважити, що у BPF програма типу XDP, мережеві пакети приходять у вигляді сирих і неопрацьованих Ethernet пакетів [6].

Блок-схема опрацювання мережевого пакету (тобто дана послідовність дій виконується на кожному пакеті, що прибуває на визначений мережевий інтерфейс) зображена на рис. 2. Лічильники та додаткові значення зберігаються у BPF мапах.

Блок-схема алгоритму функціонування частини програмного засобу із користувацького простору для завантаження BPF та налаштування комунікації із нею, містить такі кроки:

Крок 1. Зчитування номеру мережевого інтерфейсу для прослуховування.

Крок 2. Завантаження програми BPF у простір ядра.

Крок 3. Ініціалізація всіх необхідних мап для передачі даних.

Крок 4. Прикріплення завантаженої BPF програми до визначеного мережевого інтерфейсу.

Крок 5. Зчитування із мап отриманих даних.

Крок 6. Опрацювання отриманих даних і визначення актуальної статистики.

Крок 7. Виведення визначеної статистики на консоль. Перехід на крок 5.

Розроблена структура програмного засобу ґрунтується на модульному принципі, що дає змогу швидко вдосконалювати систему.

Математичне забезпечення системи. В процесі дослідження розроблено ряд моделей. Зокрема, для дослідження динаміки функціонування програмного засобу побудовано моделі на підставі теорії мереж Петрі [4], [8], [21] які описують з використанням наступного виразу:

$$M_{odel_{MP}} = (P, T, F, M_0), \quad (1)$$

де: P – матриця позицій; T – множина переходів; F – множина вхідних і вихідних дуг; M_0 – початкове маркування простої мережі Петрі.

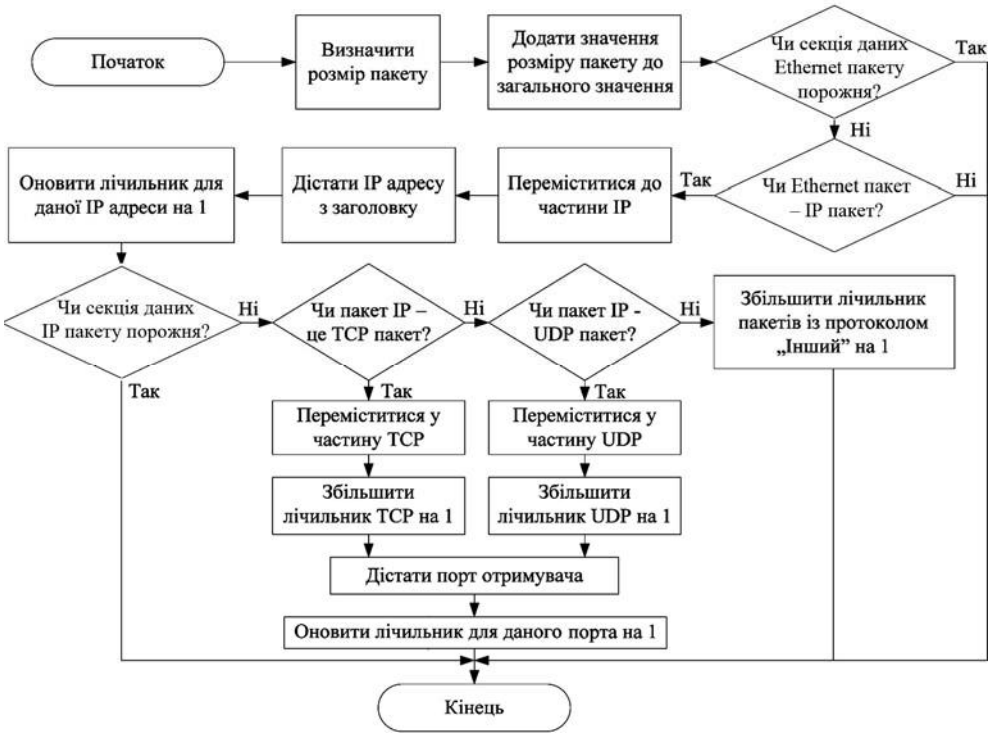

Рис. 2. Блок-схема алгоритму опрацювання прибулого Ethernet пакету / Block diagram of the algorithm for processing an incoming Ethernet packet

Внаслідок виконаного дослідження динаміки функціонування програмного засобу, будуємо граф досяжності станів, який можна описати таким виразом [19]:

$$G = (S, L), \quad (2)$$

де: S – множина позицій в яких може перебувати досліджувана система; L – множина зав'язків.

Граф (2) є орієнтованим. Напрям показує, з якого стану система може перейти в інший. Окрім цього, моделі на підставі теорії мереж Петрі дають змогу дослідити розроблювану систему на наявність тупиків, живучість, обмеженість та інші параметри.

Процес побудови системи передбачає тестування розробленого засобу. Для цього синтезовано імітаційну модель, яка містить три основні складові, що можна описати з допомогою наступного кортежа:

$$I_M = \langle S_{kp}, M_{\text{іміт}}, R_{ez} \rangle, \quad (3)$$

де: S_{kp} – модель, яка генерує вхідні дані для імітаційної моделі на підставі розробленого скрипта; $M_{\text{іміт}}$ – розроблений програмний засіб; R_{ez} – отримані результати.

Розроблена імітаційна модель дає змогу дослідити правильність та коректність роботи розроблюваного програмного засобу.

Розроблені моделі дають змогу дослідити, ще на етапі розроблення, правильність та коректність функціонування програмного засобу.

Особливості розроблення програмного забезпечення системи. Загальна структура програмного засобу визначають схемою, яка вказана на рис. 1. З рисунка слідує, що програмний засіб ділиться рівноцінно на дві частини:

Програмна частина, яка виконується на машині сервісу, який ми хочемо моніторити. BPF програма виконується у просторі ядра (захищений режим) та реалізована на мові програмування C [16]. Користувацька програма для збирання агрегованих даних і виконання серверу для відправки цих даних реалізована на мові програмування C++ [20].

Інша програмна частина виконується на машині користувача. Ця частина приймає агреговані дані для подальшої їх візуалізації та малювання графіків. Програмна частина, що виконується на машині користувача, реалізована на Python [9] та JavaScript [5].

Розроблений скрипт для генерування тестового мережевого трафіку не є частиною основного програмного засобу, проте допомагає у тестуванні основної частини розробленого програмного засобу.

Програма частина для машини сервісу обов'язково має виконуватися на операційній системі Linux, оскільки наразі тільки ОС Linux підтримує таку функцію. До того ж, більшість розробників використовують Linux як операційну систему під розроблений сервіс, через можливість детальної конфігурації виконання та безпеку. Розроблення та тестування відбувалися на операційній системі Arch Linux версії 5.10.3, яка встановлена на віртуальну машину.

Оскільки зазвичай розробники не мають прямого доступу до машини сервісу, а доступ виконується через консоль інструментами SSH, для програмної частини, виконуваній на цій машині нема потреби реалізовувати графічний інтерфейс. Саму програмну частину можна запустити наступною командою:

sudo./monitorBX -i I

Прапорець "i" у вищезазначеній команді, відповідає за передачу номеру мережевого інтерфейсу, який ми хочемо прослуховувати.

При виведенні інформації про опрацьований вхідний мережевий трафік, використано такі поля: мітка часу виводу; навантаження мережевого трафіку (у байтах / секунду); кількість викинутих пакетів; кількість пакетів переданих для подальшого опрацювання; кількість пакетів із протоколом TCP; кількість пакетів із протоколом UDP; кількість пакетів із протоколом ICMP; кількість пакетів із іншим протоколом; кількість унікальних IP адрес відправника та кількість унікальних портів отримувача.

Приклад виведення вищезазначених даних можна побачити на рис. 3. Очевидно, що виведені дані у консоль є не найкращим способом для репрезентування подій, що відбуваються в комп'ютерній мережі, тому користувач має можливість переглянути візуалізовані дані, а саме: переглянути графіки, які будує програмний засіб на підставі переданих даних і панель загальної статистики із логами. До того ж, користувач має можливість візуально порівняти поточні значення із історичними значеннями на попередніх 20 ітераціях.

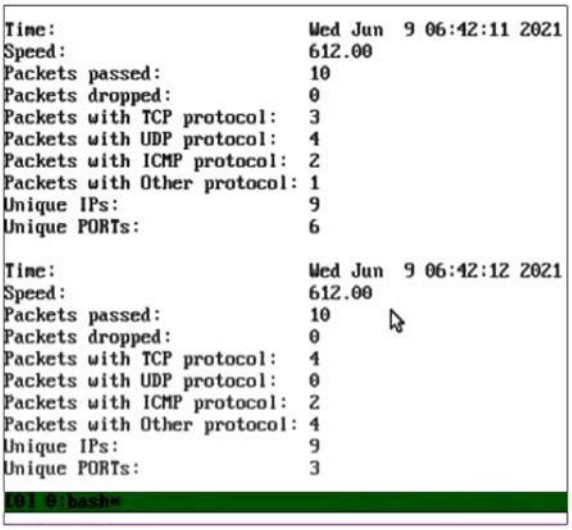

Рис. 3. Інтерфейс користувацької програми для виведення загальної статистики мережевого трафіку у консолі машини сервісу / User program interface for displaying general network traffic statistics in the console of the service machine

Розроблення скрипту для генерування мережевого трафіку. Після опису розроблення програмної частини системи, надамо інформацію про розроблення скрипта для імітаційної моделі. Для того, щоб можна було якісно протестувати розроблений програмний засіб, потрібно генерувати мережевий трафік для машини сервісу. Для цього реалізований скрипт на мові Python для генерування послідовності мережевих пакетів із IP адресою відправника, номером порта отримувача та протоколом, який обирається випадково. Ці пакети створювалися за методологією підміни IP адреси, і у подальшому відправляються на інтерфейс машини сервісу.

Розроблений скрипт запускаємо за такою командою:

python emulate_network.py [*-h*] [*-s S*] *dst*

де: *dst* – IP адреса машини сервісу, яка тестується; *h* – прапорець для допомоги; *s* – прапорець для визначення паузи між відправкою згенерованих мережевих пакетів (визначають у секундах).

Приклад виводу скрипта зображено на рис. 4.

```
Sent 1 packets.
Source IP: 170.82.31.155  Protocol: UDP   Destination port: 564.
Sent 1 packets.
Source IP: 53.162.67.6    Protocol: UDP   Destination port: 384.
Sent 1 packets.
Source IP: 242.79.225.169 Protocol: Other Destination port: 137.
Sent 1 packets.
Source IP: 190.22.48.68   Protocol: UDP   Destination port: 100.
Sent 1 packets.
Source IP: 39.203.128.186 Protocol: UDP   Destination port: 346.
Sent 1 packets.
Source IP: 91.5.248.6     Protocol: UDP   Destination port: 927.
```

Рис. 4. Вивід скрипта для генерування мережевого трафіку / Output of the script for generating network traffic

Розроблений скрипт дає змогу згенерувати вхідні дані для імітації мережі та провести тестування розробленого програмного засобу.

Обговорення результатів дослідження. Розроблений програмний засіб дає змогу відобразити на графіках ряд параметрів трафіку, а саме: Traffic load – показує навантаження мережевого трафіку протягом часу у байтах/секунду; Protocols – порівняння кількості пакетів із протоколами TCP, UDP, ICMP та іншими; Packets – порівняння кількості пакетів, які викидаються та кількістю, яка допускається для подальшої оброблення та Number of unique – порівняння у часу кількості унікальних IP адрес та портів.

Окрім цього, програмний засіб містить панель загальної статистики. Загальна статистика містить статистику за весь період виконання програмного засобу та за останній день. Ця статистика містить такі дані: Passed packets – кількість пакетів передана для подальшого опрацювання; Dropped packets – кількість викинутих пакетів; Average transfer rate (bytes/s) – середнє значення мережевого потоку за визначений час.

Наведена загальна статистика дає змогу порівняти користувачеві чи загалом сервіс набуває популярності у використанні (порівнюючи поточні значення із середніми протягом усього часу запуску програмного засобу або протягом останнього дня) або просто спостерігати за більш загальною статистикою, яка враховує довгий період, а не тільки останні 20 ітерацій, які зображені на графіках.

Приклад візуалізованих графіків зображено на рис. 5 та бокової панелі із статистикою – на рис. 7.

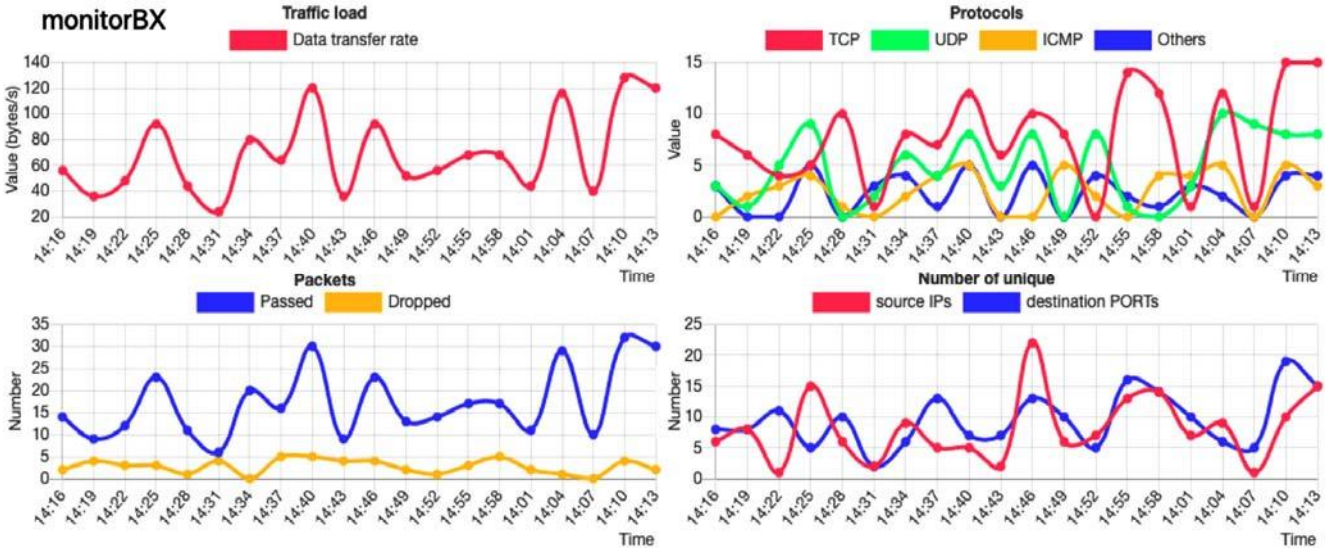

Рис. 5. Інтерфейс із візуалізацією даних статистики мережевого трафіку / Interface with visualization of network traffic statistics data

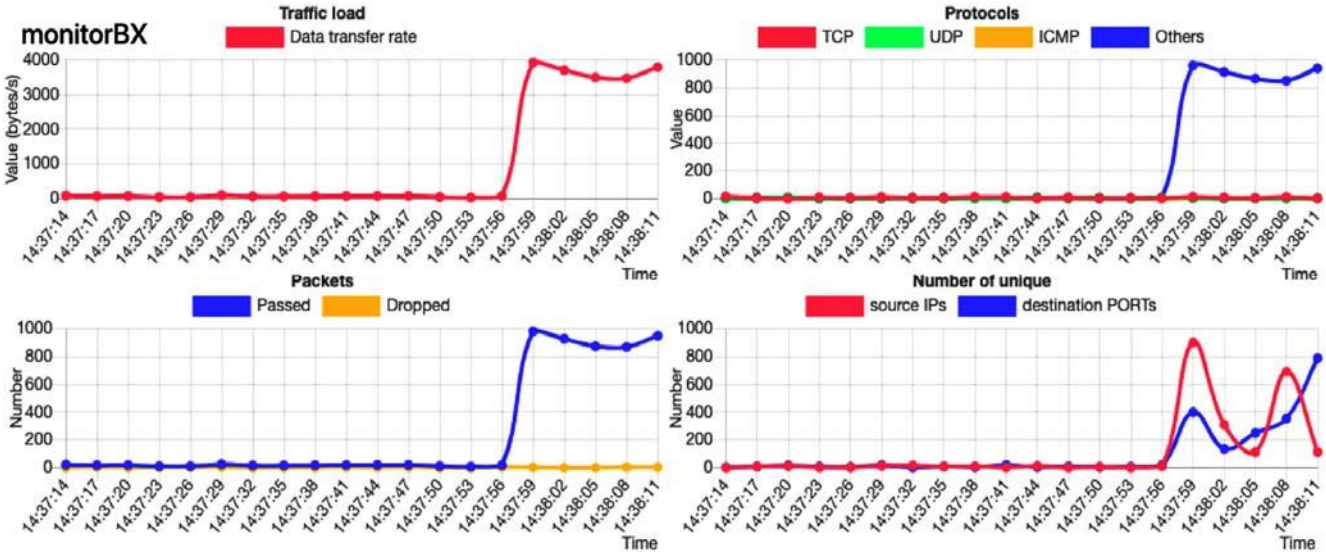

Рис. 6. Приклад меню програмного засобу з параметрами різкого скачку / An example of a menu of a software tool with parameters of a drastic increase of network traffic flow

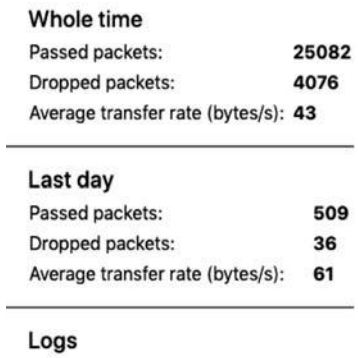

Рис. 7. Приклад вихідних даних на боковій панелі із статистикою мережевого трафіку за тривалий період часу / Example sidebar output with network traffic statistics over a long period of time

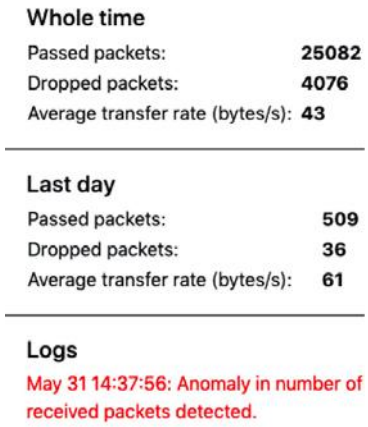

Рис. 8. Приклад попереджувального повідомлення, який видасть програмний засіб на боковій панелі / An example of a warning message that the software tool will issue on sidebar dashboard

При аналізі поточних значень трафіку та порівнюючи їх із середніми значеннями протягом тривалого часу, програмний засіб дає змогу визначити певні аномалії. Наприклад надто стрімке зростання потоку мережевого трафіку. При виявленні таких аномалій, користувач побачить попереджуюче повідомлення у логах бокової панелі. Приклад відповідного повідомлення зображено на рис. 6 та рис. 8. Завдяки таким попереджувальним повідомлення, користувач може вчасно зреагувати та прийняти заходи при DDOS атаках.

Отже, за результатами виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – полягає у вдосконаленні імітаційної моделі моніторингу трафіку комп'ютерної мережі з допомогою унікального інструмента Linux – фільтра пакетів Берклі та розробленні моделей на підставі теорії мереж Петрі для дослідження динаміки функціонування побудованої системи.

Практична значущість результатів дослідження – полягає у розробленні структури та алгоритмів роботи програмного засобу, програмній реалізації засобу з використанням фільтра пакетів Берклі, який забезпечує автоматизацію збирання, збереження та опрацювання даних про вхідний трафік в комп'ютерних мережах, відображає загальну статистику, програмна реалізація скрипта та отримані результати. Розроблений програмний засіб дає змогу збирати дані в автоматичному режимі про вхідний трафік ще на ранніх стадіях опрацювання пакету, та відображатиме статистику із ілюстративними графіками, де адміністратор роботи сервісу, зможе ідентифікувати підозрілу поведінку трафіку та застосувати необхідні дії для уникнення шкідливих дій щодо сервісу.

Висновки/Conclusions

Розроблено алгоритм та структуру системи моніторингу трафіку комп'ютерної мережі з використанням фільтра пакетів Берклі, яка ґрунтується на модульному принципі, що дає змогу швидко модифікувати систему.

Розроблено моделі на підставі теорії мереж Петрі, які дають змогу дослідити динаміку функціонування розробленого програмного засобу та відповідність вимогам ПЗ.

Розроблено імітаційну модель та скрипт для генерування параметрів трафіку комп'ютерної мережі, яка дає змогу реалізувати операцію тестування програмного засобу.

Розроблено програмне забезпечення системи моніторингу трафіку комп'ютерної мережі з використанням фільтра пакетів Берклі, яке використовує мови C, C++ та Python, що забезпечує збір, збереження та опрацювання даних трафіку комп'ютерної мережі та подання результатів дослідження у формі зручній для користувача.

Наведено результати дослідження потоку даних в комп'ютерній мережі, що підтверджує правильність та коректність функціонування розробленого програмного засобу.

References

[1] Cai, W., Song, X., Liu, C., Jiang, D., & Huo, L. (2022). An Adaptive and Efficient Network Traffic Measurement Method Based on SDN in IoT. In Proceedings of the International Conference on Simulation Tools and Techniques, Istanbul, Türkiye, 20-21 October 2022, Springer: Cham, Switzerland, 64-74. Retrieved from: https://doi.org/10.1007/978-3-030-97124-3_6

[2] Calavera, D., & Fontana, L. (2020). Linux Observability with BPF. 1005 Gravenstein Highway North, Sebastor, CA 96472: O'Relly Media, Inc.

[3] Cilium. (2022). Cilium documentation on BPF. Retrieved from: https://docs.cilium.io/en/stable/bpf/

[4] Gonçalves, E. M. N., Machado, R. A., Rodrigues, B. C., & Adamatti, D. (2022). CPN4M: Testing Multi-Agent Systems under Organizational Model Moise+ Using Colored Petri Nets. *Appl. Sci.*, *12*, 5857. https://doi.org/10.3390/app12125857

[5] Haverbeke, M. (2018). Eloquent JavaScript, 3rd edition, No Starch Press. Retrieved from: https://eloquentjavascript.net/

[6] Horodetska, O. S., Hykavy, V. A., & Onyshchuk, O. V. (Ed). (2017). Computer networks. Vinnytsia: VNTU Publishing House. [In Ukrainian].

[7] Java, G. P. (2002). IPTraf User's Manual. Retrieved from: iptraf.seul.org/2.6/manual.html

[8] Kim, I., & Xu, S. (2019). Bus voltage control and optimization strategies for power flow analyses using Petri net approach. Int. J. Electr. Power Energy Syst., 112, 353-361. https://doi.org/10.1016/j.ijepes.2019.05.009

[9] Kostyuchenko, A. O. (2020). Basics of Python programming. Chernihiv: FOP Balykin S. M. [In Ukrainian].

[10] Mo, L., Lv, G., & Wang, B. (2022). A Fine-Grained Network Congestion Detection Based on Flow Watermarking. Appl. Sci., 12, 8094. https://doi.org/10.3390/app12168094

[11] Nayak, P., & Knightly, E. W. (2022). Virtual speed test: An ap tool for passive analysis of wireless lans. *Comput. Commun*, 192, 185-196. https://doi.org/10.1016/j.comcom.2022.05.031
[12] NetLog. (2022). Official site. Retrieved from: https://netlog.sourceforge.net/
[13] OpenSSH. (2022). Official site. Retrieved from: www.openssh.com
[14] Prototype Kernel. (2022). Kernel documentation for eBPF maps. Retrieved from: https://prototype-kernel.readthedocs.io/en/latest/bpf/ebpf_maps.html
[15] Roland-riegel. (2018). Nload. Official site. Retrieved from: www.roland-riegel.de/nload/
[16] Shpak, Z. Ya. (2011). Programming in the language of S. Lviv: Publishing House of Lviv Polytechnic. [In Ukrainian].
[17] Silver Moon. (2020). 18 Commands to Monitor Network Bandwidth on Linux server. Retrieved from: https://www.binarytides.com/linux-commands-monitor-network/
[18] Tcpdump & libpcap. (2022). Official site. Retrieved from: www.tcpdump.org
[19] Teslyuk, T. V. (2018). The analysis of the dynamics of the functioning of multilevel systems using models based on hierarchical petri networks. *Scientific Bulletin of UNFU, 28*(8), 149-154. https://doi.org/10.15421/40280830
[20] Trofymenko, O. G., Prokop, Y. V., Shvaiko, I. G., Bukata, L. M., Kosyreva, L. A., Leonov, Y. G., & Yasinsky, V. V. (2010). C++. Fundamentals of programming. Theory and practice. Odesa: "Fenix" Publishing House. [In Ukrainian].
[21] Wang, Y., Yin, X., Yin, X., Qiao, J., & Tan, L. (2022). A Petri Net-Based Power Supply Recovery Strategy for the Electric Power System of Floating Nuclear Power Plant. *Appl. Sci.*, *12*, 9026. https://doi.org/10.3390/app12189026
[22] Wilson, M. (2021). 11 Best Network Monitoring Tools Software of 2021. Retrieved from: https://www.pcwdld.com/best-network-monitoring-tools-and-software

I. G. Tsmots, S. V. Tesliuk

Lviv Polytechnic National University, Lviv, Ukraine

MODELS AND TOOLS OF THE AUTOMATED SYSTEM FOR COMPUTER NETWORK TRAFFIC INVESTIGATION USING BERKELEY PACKET FILTER

An approach for automating the monitoring and analysis of incoming network traffic in large-scale computer networks is proposed in the paper. The authors suggest using the Linux Berkeley Packet Filter tool to automate traffic analysis in computer networks. The software structure is developed, which includes two main parts: the service machine and the user machine, it is based on the modular principle, which allows for rapid improvement and modernization of the system. The main algorithms for software functionality are built, namely: the algorithm for processing network packets using the Berkeley Packet Filter tool, and the algorithm of the user-space program for loading the Berkeley Packet Filter program to kernel space and setting up communication with it. A study model of program functioning dynamics based on the Petri net theory has been developed. As a result of the application of models based on the Petri net in the software development process, the system works correctly, all states are accessible, and there are no dead ends. A simulation model of the application of the Berkeley Packet Filter tool for the automation of computer network traffic analysis was designed, and the script was created for testing the developed software system. Implemented Python script generates a flow of network packets with random values in the sender IP address, receiver port number, and protocol. These packets, created by the IP address spoofing methodology, later are sent to the service machine's network interface. The developed computer network traffic monitoring software, that uses the Berkeley Packet Filter tool and is implemented in C, C++, and Python programming languages, provides collecting and processing of computer network traffic data. The output of the analysed results is displayed in a user-friendly form. The development and testing of the created software were carried out on the operating system Arch Linux version 5.10.3, which was previously installed on a virtual machine. The results of traffic testing in computer networks in different modes of normal operation and during DDoS attacks are given. In particular, an example of sidebar output with network traffic statistics over a long period, an example of output with network traffic spike parameters, and an example of a warning message, that the sidebar dashboard will show, are presented.

Keywords: network packet processing algorithm; the algorithm of the functioning of the software tool; traffic research automation; Petri net; simulation model; DDoS attack.

Інформація про авторів:

Цмоць Іван Григорович, д-р техн. наук, професор, кафедра автоматизованих систем управління.
Email: ivan.g.tsmots@lpnu.ua; https://orcid.org/0000-0002-4033-8618

Теслюк Софія Василівна, магістр кафедри автоматизованих систем управління. **E-mail:** sofiia.tesliuk.mknus.2021@lpnu.ua

Цитування за ДСТУ: Цмоць І. Г., Теслюк С. В. Моделі та засоби автоматизованої системи дослідження трафіку комп'ютерних мереж з використанням фільтра пакетів Берклі. *Український журнал інформаційних технологій.* 2022, т. 4, № 2. С. 61–67.

Citation APA: Tsmots, I. G., & Tesliuk, S. V. (2022). Models and tools of the automated system for computer network traffic investigation using Berkeley Packet Filter. *Ukrainian Journal of Information Technology, 4*(2), 61–67. https://doi.org/10.23939/ujit2022.02.061