Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему: «Метод виявлення DDoS-атак»

08-20.МКР.007.00.000 ПЗ

Виконав: студент 2 курсу, групи 1БС-18м
спеціальність 125 – Кібербезпека

_____ Ковальов В. А.

Керівник: к. т. н., доц. каф. ЗІ

_____ Дудатьєв А. В.

Рецензент
к. т. н., доц. каф. ОТ

_____ Крупельницький Л. В.

Вінниця – 2019 року

АНОТАЦІЯ

Магістрська кваліфікаційна робота присвячена розробці методу виявлення DDoS-атак. Для практичної реалізації методу виявлення DDoS-атак було розроблено було розроблено програмний засіб та протестовано метод на реальній комп`ютерній мережі.

Для успішної розробки методу проведено дослідження атак, їх статистичний аналіз, аналіз різних видів атак та аналіз різних видів мережевих пакетів, обґрунтовано доцільність розробки даного методу. У даній роботі розроблено ряд схем і алгоритмів, здійснено програмну реалізацію. Метод та програмний засіб перевірено на коректність роботи та доведено ефективність методу виявлення атак.

ABSTRACT

Master's qualification work is devoted to the development of the method of detection of DDoS-attacks. For practical implementation of the DDoS attack detection method, a software tool was developed and the method was tested on a real computer network.

For the successful development of the method the study of attacks, their statistical analysis, analysis of different types of attacks and analysis of different types of network packets, the feasibility of developing this method was done. In this work, a number of circuits and algorithms are developed, software implementation is made. The method and software were tested for accuracy and the effectiveness of the attack detection method was proven.

Вінницький національний технічний університет

Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Захисту інформації
Освітньо-кваліфікаційний рівень магістр
Спеціальність 125 – Кібербезпека

ЗАТВЕРДЖУЮ
д.т.н., проф. зав. кафедри ЗІ
_____В. А. Лужецький
“____” _____2019 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ковальову Володимиру Андрійовичу

1. Тема роботи: «Метод виявлення DDoS-атак»

керівник роботи: Дудатьєв Андрій Веніамінович, к.т.н., доц. кафедри ЗІ,
затверджені наказом № 254 ректора ВНТУ від 02.10.2019 р.
2. Строк подання студентом роботи 12 грудня 2019 р.
3. Вихідні дані до роботи:
   - Мови розробки: JavaScript.
   - Гібридний метод виявлення DDoS-атак.
   - Програмний засіб для реалізації методу.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. 1. Аналіз та обгрунтування теми. 2. Розробка методу виявлення DDoS-атак. 3. Тестування програмного засобу. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу
Схема тестової мережі (плакат А4). Статистична діаграма DDoS-атак за видами (плакат А4). Схема ланцюжка відповідальностей (плакат А4). Схема послідовності взаємодії програмного засобу (плакат А4). Лічильник з обмеженням по ресурсам (плакат А4). Таблиця порівняння розробленого методу виявлення з аналогами (плакат А4). Схема алгоритму роботи 4ЕКС (плакат А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Дудатьєв А.В., к.т.н., доц. каф.ЗІ		
2	Дудатьєв А.В., к.т.н., доц. каф.ЗІ		
3	Дудатьєв А.В., к.т.н., доц. каф.ЗІ		
4	Мацкевічус С. С. ст. вик. каф. ЕПВМ		

7. Дата видачі завдання _____ 2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.19 – 04.09.19	
2	Аналіз літературних джерел за напрямком бакалаврської дипломної роботи	05.09.19 – 15.09.19	
3	Науково-технічне обгрунтування	16.09.19 – 21.09.19	
4	Розробка технічного завдання		
5	Розробка рішень	23.09.19 – 30.09.19	
6	Практична реалізація, моделювання, експериментування, результати	30.09.19 – 10.09.19	
7	Розробка розділу економічного обгрунтування доцільності розробки	14.10.19– 10.11.19	
8	Аналіз виконання ТЗ, висновки	11.11.19 – 17.11.19	
9	Оформлення пояснювальної записки	18.11.19 – 24.11.19	
10	Попередній захист МКР	25.11.19 – 30.11.19	
11	Перевірка магістерської роботи на наявність плагіату	28.11.19 – 01.12.19	
12	Виправлення зауважень, підготовка ілюстративного матеріалу	02.12.19 – 10.12.19	
13	Представлення МКР до захисту, рецензування	11.12.19 – 14.12.19	
14	Захист МКР	16.12.19 – 20.12.19	

Студент _____ Ковальов В. А.

(підпис)

Керівник роботи _____ Дудатьєв А. В.
(підпис)

ЗМІСТ

ВСТУП

Сучасна людина володіє різноманітною інформацією. Інформація може бути як даними про акторів улюблених фільмів та учасників улюблених гуртів, так і інформація про будь-які події, які відбуваються на виробництві.

Оскільки сучасна інформація має властивість прив’язувати до себе певну матеріальну цінність, відповідно почали з’являтись люди, які посягають на цілісність інформації, що зберігається та серверів на яких ця інформація знаходиться.

На сьогоднішній день ми не можемо уявити своє життя без комп’ютерів. Кожен комп’ютер у світі є частиною глобальної мережеві Інтернет. Щодня з метою нанести певних збитків або призвести до відмови багатьох серверів відбуваються тисяці мережевих атак. Найпоширенішим видом мережевих атак є DDoS атаки, які призводять до тимчасової або повної відмови забезпечення певної послуги або перевантаження серверів. Даний вид атак наносить нищівної шкоди великим підприємствам, що працюють в мережі Інтернет, якщо зловмисникам вдалося зупинити роботу серверів бодай на пів години.

Кожен зловмисник, який намагається поцупити чи ушкодити інформацію, яка йому не належить або він не має права доступу до неї, повинен бути притягнутий до кримінальної відповідальності за статтею 361 Кримінального кодексу України, яка говорить, що несанкціоноване втручання в роботу електронно-обчислювальних машин (комп'ютерів), автоматизованих систем, комп'ютерних мереж чи мереж електрозв'язку, що призвело до витоку, втрати, підробки, блокування інформації, спотворення процесу обробки інформації або до порушення встановленого порядку її маршрутизації, карається штрафом від шестисот до тисячі неоподатковуваних мінімумів доходів громадян або обмеженням волі на строк від двох до п'яти років, або позбавленням волі на строк до трьох років, з позбавленням права обіймати певні посади чи займатися певною діяльністю на строк до двох років або без

такого та з конфіскацією програмних та технічних засобів, за допомогою яких було вчинено несанкціоноване втручання, які є власністю винної особи [1].

Тому було прийнято рішення, як фахівця з захисту інформаційних і комунікаційних систем розробити метод для виявлення DDoS-атак. Актуальністю даної теми є те, що методи виявлення подібного виду атак часто мають похибки і не точності в роботі і самі є причиною негативних наслідків в роботі комп`ютерних мереж. Об`єктом дослідження в даній роботі стали DDoS-атаки та процеси виявлення даного виду атак.

Метою магістрської дипломної роботи є покращення методу виявлення мережевих атак.

Для досягнення мети необхідно розв’язати такі задачі:

- проаналізувати види мережевих атак;
- зроботи статистичний аналіз DDoS-атак;
- зробити аналіз мережевих протоколів;
- розробити метод виявлення атак;
- протестувати розроблений метод.

Науковою новизною даного методу є його покращення відносно аналогів за рахунок зменшення можливої похибки під час виявлення атак.

1 АНАЛІЗ ТА ОБГРУНТУВАННЯ ТЕМИ

1.1 Схема DDoS-атаки

DDoS-атака полягає у скоординованому посиланні величезної кількості помилкових запитів від безлічі комп'ютерів на ресурс що атакується. В результаті атакований сервер витрачає всі свої ресурси на обслуговування цих запитів і стає практично недоступним для звичайних користувачів. Ситуація ускладнюється тим, що користувачі комп'ютерів, з яких направляються помилкові запити, можуть навіть не підозрювати про те, що їхні комп'ютери використовуються спеціальними троянами. Найчастіше зловмисники при проведенні DDoS-атак використовують трирівневу архітектуру. Простежити таку структуру в зворотному напрямку і виявити адресу вузла, який організував атаку, практично неможливо. Максимум того, що може атакований ресурс це визначити адреси атаки, спеціальні заходи в кращому разі приведуть до центру управління ботнетом, але зазвичай це заражені комп'ютери і власники не підозрюють про свою участь в атаці. Кожного дня у світі відбуваються тисячі DDoS-атак з різним масштабом (Рис. 1.1) [2].

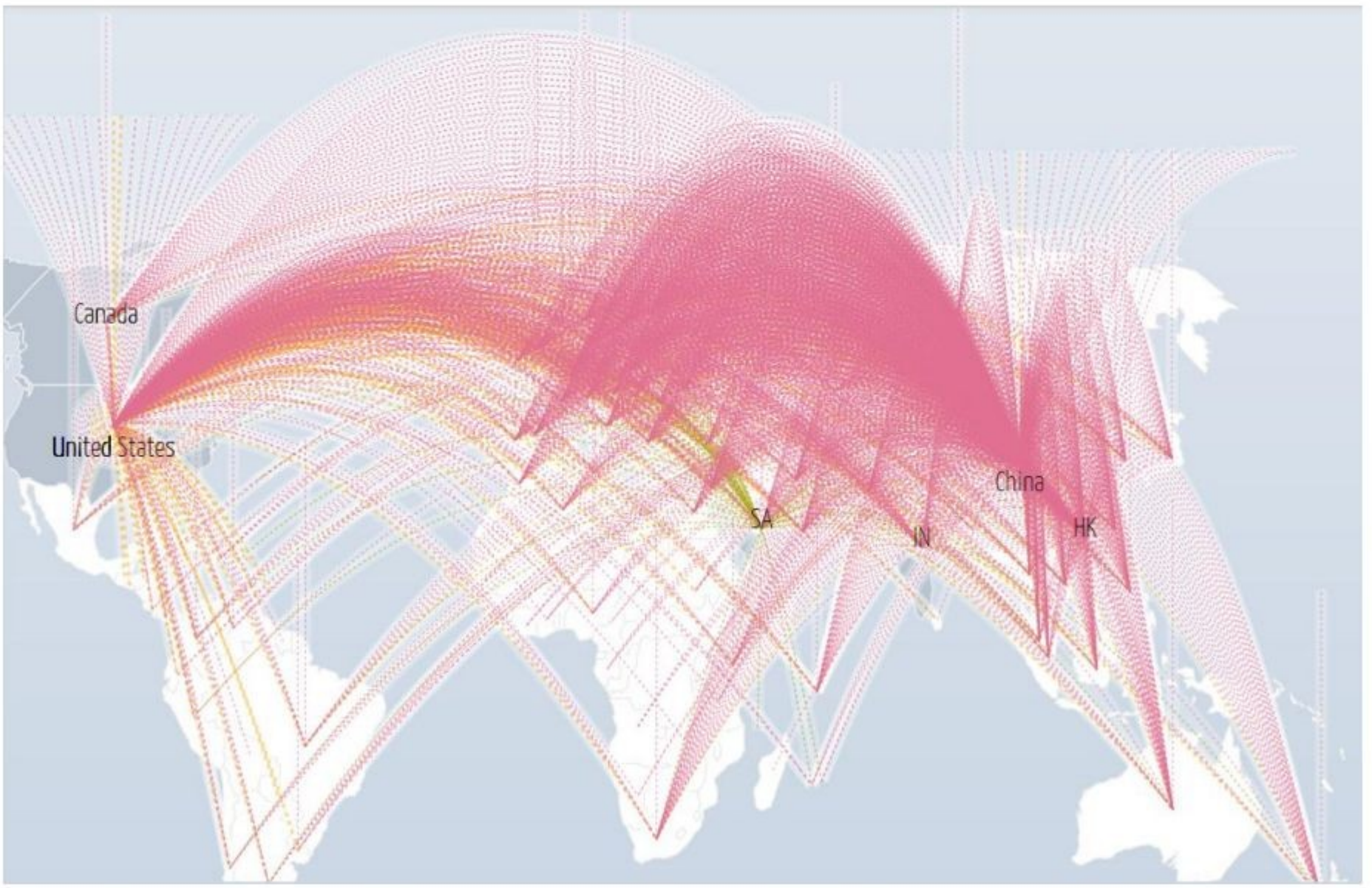


Рисунок 1.1 – Карта зареєстрованих DDoS-атак в реальному часі

В таблиці 1.1 показані найпопулярніші та найвідоміші засоби для здійснення DDoS-атак та їх основні специфікації.

Таблиця 1.1 – Засоби DDoS-атаки та їх специфікації

Засоби DDoS-атаки	Зв'язок Intruder-to-master	Зв'язок Master-to-daemon	Зв'язок Daemon-to-master
Trinoo	27665/tcp	27444/udp	31335/udp
TFN	ICMP Echo/Echo Reply	ICMP Echo Reply	ICMP Echo/Echo Reply
Stacheldraht	16660/tcp	65000/tcp	ICMP Echo Reply
Trinity	6667/tcp	6667/tcp або 33270/tcp	
Shaft	20432/tcp	18753/udp	20433/udp

1.2 Основні механізми DDOS-атак

Існує декілька основних видів DDoS-атак, розглянемо їх нижче:

– ICMP SMURF

Найбільш розповсюджений метод DDoS-атак. Принцип його роботі доволі простий. Зловмисник, змінюючи адресу джерела, посилає пакет ICMP Echo Request (відомий як пінг (ping)) до окремих комп'ютерів, як правило, тих що входять до складу мережі ddosnet). Ці комп'ютери в свою чергу відповідають пакетом ICMP Echo Reply, відправляючи його на IP адресу вказану хакером як джерело. Часто для підсилення атаки використвуються локальні мережі (LAN) з ввімкненою опцією направленого широкомовного повідомлення (directed broadcast), у відповідь на пінг з кожного комп'ютера в складі мережі. Наприклад на один запит буде відправлено 100 відповідей.[3]

– SYN FLOOD

Також один з найчастіших методів DDoS-атак. Для того щоб зрозуміти схему його діяльності, потрібно знати, як саме встановлюється з'єднання в протоколі TCP (Transport Control Protocol). Syn Flood атака базується на створенні, так званих,

напіввідкритих сесіях. Один з агентів посилає SYN пакет, у відповідь на який сервер вимушений відповідати SYN та ACK пакетами. Ціллю такої DDoS-атаки являється не перевантаження каналу, а припинення зв'язку між адміністратором і сервером, а також максимальне завантаження пам'яті сервера, через що можуть виникнути проблеми з доступом до сервісів або ж повна відмова у постачанні послуг.[4]

– TCP SYN Scan

Зловмисник використовує сканування SYN для визначення стану портів на віддаленій цілі. Сканування SYN - найпоширеніший тип сканування порту, який використовується через його величезні переваги та недоліки. В результаті нападники новачки, як правило, надмірно покладаються на сканування SYN під час виконання розвідки системи. Як метод сканування основними перевагами SYN Scan є його універсальність та швидкість. RFC 793 визначає необхідну поведінку будь-якого пристрою TCP / IP, оскільки вхідний запит на отримання з'єднання починається з пакета SYN, який, у свою чергу, повинен мати пакет SYN / ACK з служби прийому. Таким чином сканування SYN працює проти будь-якого стеку TCP. На відміну від сканування TCP Connect, за допомогою якого можна сканувати тисячі портів за секунду. Цей тип сканування зазвичай називають скануванням "напіввідкритим", оскільки він не завершує тристоронній рукостискання. Швидкість сканування надзвичайно велика, тому що час не витрачається для завершення рукостискання або зривання з'єднання. Сканування TCP SYN також може негайно виявити 3 з 4 важливих типів станів портів: відкритий, закритий та відфільтрований. Коли SYN надсилається на відкритий порт і нефільтрований порт, буде сформовано SYN / ACK. Ця методика дозволяє зловмисникові сканувати через загальну конфігурацію довірених брандмауерів, так як сегменти TCP SYN для нового з'єднання будуть доступні практично для будь-якого порту. Коли пакет SYN надсилається на закритий порт, створюється RST, що вказує на те, що порт закритий. Якщо SYN-сканування до певного порту не дає відповіді, або якщо запит

викликає недоступні помилки типу ICMP Type 3, порт фільтрується. Сканування TCP Connect має такі характеристики:

1. Швидкість: сканування TCP SYN відбувається швидко, порівняно з іншими типами сканування.
2. Непомітність: сканування TCP SYN - секретне, а виявлення SYN сканування загрожує помилковими діями.
3. Відкритий порт: виявляє, що порт відкритий через успішну SYN / ACK до SYN.
4. Закритий порт: виявляє, що порт закритий через успішний RST до SYN.
5. Відфільтрований порт: без відповіді або повідомлення ICMP - це наявність фільтра.
6. Нефільтрований порт: неможливо розрізняти повністю відфільтрований порт та порт, що не фільтрується.

SYN-сканування відбувається швидко і надає зловмиснику велику кількість інформації. Первинний недолік полягає в тому, що для сканування SYN потрібна можливість доступу до "сирих сокетів" для створення пакетів.

– UDP FLOOD

Даний метод базується на використанні UDP протоколу і використовується для виклику найбільш можливого навантаження каналу системи, що атакується. Метод UDP Flood не являється атакою, яка використовує недоліки протоколу, а базується на додатках, які працюють в рамках протоколу UDP. Два сервіси, що є найбільш частими мішенями UDP Flood атаки – це системи DNS і Chargen.

– UDP Flood DNS

Метою DDoS-атаки на DNS сервер є порушення перетворення символьних адрес в IP адреси. Це може паралізувати деякі послуги і заважати взаємодії користувача з мережею інтернет. Атака служб DNS є важкою і часто не приносить бажаного ефекту. Хакери використовують два варіанти атаки на сервери. Перший відноситься до системи імен визначеної під мережі. В даному випадку зловмисник посилає більшості DNS серверів запит про домен жертви, який

генерує відносно високу активність і тим самим може негативно відобразитись на роботі атакованого сервера. Суть другого варіанту атаки – відправка жертві великої кількості запитів про IP адресу неіснуючого домену. В даному випадку відповідь від DNS сервера значно перевищує об'єм запитів відправлених хакером, що може призвести до перевантаження каналів, що використовує жертва.

– UDP Flood Chargen

Chargen (скорочено від Character Generator) це сервіс, який генерує набір випадкових чисел від 0 до 512 різної довжини. Він слугує для діагностичних цілей і дозволяє визначити причини втрати пакетів. Напад на сервіс Chargen є простим і ефективним методом DDoS-атак, заснований на відправці на нього з багатьох комп'ютерів невеликих пакетів UDP, на які послуга повинна відповісти, в результаті чого часто відбувається перевантаження з'єднання.

Існує також інший варіант нападу, коли жертва має у своїй мережі два сервери, які забезпечують одну й ту саму послугу. В такому випадку використовується метод підміни адреси, хакер видає себе за перший сервер, а потім відправляє на другий сервер набор символів. Це приводить до відповіді на перший сервер і так по колу, що створює нескінченний цикл звернень і пере посилань. Це в свою чергу призводить до непрацездатності серверів.

– HTTP FLOOD

Цей тип атаки доволі поширений при атаках на сайти, які знаходяться на сервері. Механізм HTTP Flood базується на відправці максимально можливого числа запитів для відображення документів, що зміщені на сервері. В результаті такої атаки може бути припинено надання послуг, які пов'язані з HTTP і ускладнення з доступом до сайту для клієнтів. Існує два способи проведення такої атаки, перший виконується через ddosnet мережу, а другий – через проксі-сервери. Перший метод є більш надійним і простим. Комп'ютери, що об'єднані в мережу відправляють запити на окремі визначені документи, представленні на HTTP сервері, за достатньо масштабної атаки це може призвести навіть до повного припинення забезпечення послуг. Другий метод використовується, як підсилення

атаки. Він базується на використанні великої кількості загальнодоступних проксі-серверів. Все це виконується в три етапи. Спочатку хакер посилає запит на w3cache сервера для відображення певної сторінки з ресурсів жертви. Сервери негайно з'єднуються з метою завантажити необхідні документи, а атакуючий тимчасово розриває з ними зв'язок. Цей процес повторюється до поки не буде отримано бажаного результату[5].

– Недостатня кількість ресурсів

Зловмисники використовують даний метод DoS-атак для захоплення системних ресурсів, таких оперативна і фізична пам'ять, процесорний час та інші. Зазвичай такі атаки проводяться з урахуванням того, що хакер вже володіє певною кількістю ресурсів системи. Ціллю атаки є захоплення додаткових ресурсів. Для цього не обов'язково переповнювати канал зв'зку, а достатньо просто перевантажити процесор жертви, тобто зайняти весь допустимий процесорний час.[6]

– Відправка «важких» пакетів

Нападник відправляє серверу пакети, які не заповнюють канал зв'язку(канал зазвичай доволі широкий), але витрачає весь його процесорний час. Процесор сервера, коли буде опрацьовувати їх, може не впоратись зі складними обчисленнями. Через це відбудеться збій і користувачі не зможуть отримати доступ до небхідних ресурсів.

– Переповнення сервера лог-файлами

Лог-файли сервера — це файли, в яких записуються дії користувачів мережі або програми. Некваліфікований адміністратор може неправильно налаштувати систему на своєму сервері, не встановивши визначений ліміт. Хакер скористається цією помилкою і буде відправляти пакеті великого об'єму, які згодом займуть все вільне місце на жорсткому диску сервера. Але ця атака спрацює тільки у випадку з недосвідченим адміністратором, кваліфіковані зберігають лог-файли на окремому системному диску.

1.3 Виявлення DDoS-атак

Існує думка, що спеціальні засоби для виявлення DDoS-атак не потрібні, оскільки факт DDoS-атаки неможливо не помітити. В багатьох випадках це дійсно так. Однак достатньо часто спостерігались вдалі DDoS-атаки, які були виявлені жертвами лише після 2-3 діб. Бувало, що негативні наслідки атаки (*флуд*-атаки) переходили в зайві трати на сплату перевищеного Internet-трафіку, що проявлялось лише при отриманні рахунку від Internet-провайдера. Окрім того, багато методів виявлення атак неефективні поряд із об'єктом аткаи, але є ефективними на мережевих магістральних каналах. В такому випадку вигідно встановлювати системи вивлення саме там, а не чекати, поки користувач,який попав під атаку, сам її помітить і звернеться за допомогою. До того ж для ефективної протидії DDoS-атакам необхідно знати тип, характер та інші характеристики DDoS-атак, а швидко отримати ці дані дозволяють саме служби безпеки. Вони допомогають встановити певні налаштування системи. Але визначити, чи була дана атака зроблена зловмисником, чи відмова в обслуговуванні була наслідком певної події, вони не можуть. Відповідно до правил політики безпеки, при виявленні DoS або DDoS-атаки потрібна її реєстрація для подальшого аудиту. Після того, як атака була зафіксована, може служби гарантування безпеки для деяких правок в системі та для повернення її до попереднього рівня роботи. Також для виявлення DDoS-атаки можуть використовуватися служби, не пов'язані з безпекою, наприклад, перенаправлення трафіку по іншим каналам зв'язку, підключення резервних серверів для копіювання інформації. Таким чином, засоби для виявлення і попередження DDoS-атак можуть значно різнитися в залежності від виду системи, що захищається.[7]

Методи виявлення DDoS-атак можна розділити на декілька великих груп:

- сигнатурні — базуються на якісному аналізі трафіку.
- статистичні — базуються на кількісному аналізі трафіку.

- гібридні (комбіновані) — включають в себе переваги обох вище зазначених методів

1.4 Захист від DDoS-атак

Повністю захиститися від DDoS-атак на сьогоднішній день неможливо, так як цілком надійних систем не існує. Також важливу роль відіграє людський фактор, тому що будь-яка помилка системного адміністратора, який неправильно налаштував маршрутизатор, може призвести до нищівних наслідків. Однак, не дивлячись на все це, на даний момент існує велика кількість як апаратно-програмних засобів захисту, так і організаційних методів запобігання.

Засоби протидії DDoS-атакам можна розділити на пасивні і активні, а також на превентивні и реакційні. Нижче наведений короткий список основних методів.

- Запобігання. Профілактика причин, що змушують тих або інших осіб організовувати та здійснювати DDoS-атаки. (Дуже часто кібератаки є наслідками особистих конфліктів і образ, політичних або релігійних чи інших розбіжностей у поглядах провокуючої поведінки жертви і т.д). Потрібно завчасно усувати причини DDoS-атак, після цього зробити висновки, щоб запобігти подібних атак у майбутньому.
- Зустрічні міри. Вводячи технічні або правові міри, потрібно як умога активніше діяти на джерело та організатора DDoS-атаки. Сьогодні навіть існують спеціальні фірми, які допомагають віднайти не тільки особу, яка здійснила атаку, але й самого організатора.
- Програмне забезпечення. На ринку сучасного програмного і апаратного забезпечення існує і таке, що може захистити малий і середній бізнес від слабких DDoS-атак. Ці засоби зазвичай представляють собою невеликий сервер.

- Фільтрація і блекхолинг. Блокування вихідного трафіку від атакуючих машин. Ефективність цих методі знижується по мірі наближення до об'єкта атаки і підвищується по мірі наближення до атакуючої машини. В даному випадку фільтрація може бути двох видів: використання міжмережевих екранів і списків ACL. Використання міжмережевих екранів блокує конкретний потік трафіку, але не дозволяє відділити «хороший» трафік від «поганого». ACL списки фільтрують другорядні протоколи і не торкаються до протоколів TCP. Це не зменшує швидкість роботи сервера, але це марно в тому випадку, якщо зловмисник використовує першочергові запити.[8]
- Зворотній DDOS — перенаправлення трафіку, використаного для атаки, на атакуючого. При достатній потужності атакованого сервера дозволяє не тільки успішно блокувати атаку, але й вивести з ладу сервер атакуючого.
- Усунення вразливостей. Не працює проти *флуд*-атак, для яких «вразливістю» є обмеженість тих чи інших системних ресурсів. Вана міра націлена на усунення помилок в системах і службах.
- Збільшення обсягу ресурсів. Абсолютного це не надасть, але може стати хорошою базою для використання інших видів захисту від DDoS-атак.
- Розподіл. Побудова розподілених і дубльованих систем, які не припинять обслуговувати користувачів, навіть якщо деякі їх елементи стануть недоступними через DoS-атаки.
- Ухилення. Відхилення безпосередньої цілі атаки (доменного імені або IP-адреси) подалі від інших ресурсів, які часто також є вразливими разом з самою ціллю атаки.
- Активні зворотні дії. Вплив на джерело, організатора або центр управління атакою, як технічними так і організаційно-правовими методами.
- Використання обладнання для захисту від DDoS-атак.
- Купівля сервісу по захисту від DDoS-атак. Актуально у випадку перевищення флудом пропускної можливості мережевого каналу.

Також компанія Google готова представити свої ресурси для відображення контенту вашого сайту в тому випадку, якщо сайт знаходиться під DDoS-атакою. На даний момент сервіс Project Shield знаходиться на стадії тестування, але туди можуть бути прийняті сайти деяких тематикно туда могут быть приняты сайты некоторых тематик[9]. Ціль проекту – захистити свободу слова.

1.5 Аналіз мережевих протоколів

TCP / IP - це сімейство мережевих протоколів, орієнтованих на спільну роботу. Основними протоколами, які входять в це сімейство, являются[10]:

- IP (Internet Protocol, протокол Інтернету) - забезпечує передачу даних, так званих дейтаграм, від одного вузла до іншого. При цьому кожен вузол повинен ідентифікуватися IP-адресою. Саме IP-протокол відповідає за адресацію всієї мережі за допомогою IP-адрес, так як тільки в заголовках TP-дейтаграм використовуються IP-адреси. Цей протокол є-ється ненадійним протоколом без встановлення з'єднання. Це означає, що кожна дейтаграма передається по мережі незалежно від інших, а, отже, не гарантується, що дейтаграми прийдуть в пункт призначення, і збережеться порядок їх слідування.
- ICMP (Internet Control Message Protocol, протокол керуючих повідомлень в Інтернеті) - відповідає за різні види низькорівневої підтримки протоколу IP, наприклад, передає повідомлення про труднощі маршрутизації дейтаграм і інші діагностичні повідомлення.
- TCP (Transmission Control Protocol, протокол управління передачею) - надійний протокол із установленням з'єднання. Тобто цей протокол забезпечує надійну доставку потоків даних і службу підтримки віртуальних з'єднань за рахунок використання підтверджень і повторної передачі пакетів при виникненні необхідності.

– UDP (User Datagram Protoco, протокол призначений для користувача дейтаграм) - забезпечує просту службу передачі дейтаграм конкретним додаткам на зазначеному вузлі без гарантії доставки.

На Рисунку 1.2 показано приблизне співвідношення рівнів моделі OSI і стека TCP/IP з деякими протоколами.

Рисунок 1.2 – Приблизне співвідношення рівнів моделі OSI та стеку TCP/IP з деякими протоколами

Еталонна модель OSI	Протоколи	Стек TCP/IP
Прикладной (Application layer)	HTTP, FTP, Telnet SMTP, SSL, SSH, SNMP	Прикладний (Application layer)
Представницький(Presentation layer)		
Сеансовий (Session layer)		
Транспортний (Transport layer)	TCP, UDP	Транспортний (Host-to-Host layer)
Мережевий (Network layer)	IP, ICMP, IGMP, RIP, ARP, RARP, OSPF	Міжмережевий (Internet layer)
Канальний (Data link layer)	Ethernet FDDI, ATM. PPP, SLIP, X.25,Token Ring	Доступу до мережі (Network interface layer)
Фізичний (Physical layer)		

1.6 Аналіз заголовків пакетів

– Ethernet-заголовок. На Рисунку 1.3 відображено формат Ethernet-пакету

Рисунок 1.3 - формат Ethernet-пакету

Аппаратна адреса отримувача (6 байт)	Аппаратна адреса відправника (6 байт)	Тип пакета (2 байта)
Дані		

– IP-заголовок. На Рисунку 1.4 відображено формат IP-пакету.

Рисунок 1.4 – Формат IP-пакету

Заголовск канального рівня				
Версія (4 біта)	Довжина заголовка (4 біта)	Тип сервіса (8 біт)	Загальна довжина (16 біт)	
Ідентификатор пакета (16 біт)	0	DF	MF	Зміщення фрагмента (13 біт)
Час життя (8 біт)	Протокол (8 біт)	Контрольна сумма заголовка (16 біт)		
IP-адреса відправника (32 біта)				
IP-адреса отримувача (32 біта)				
Параметри і вирівнювання (до 40 байт)				
Дані				

– TCP-заголовок. На Рисунку 1.5 відображено формат TCP-пакету.

Рисунок 1.5 – Формат TCP-пакету

IP-заголовок										
Порт відправника (16 біт)						Порт отримувача (16 біт)				
Порядковий номер (32 біта)										
Номер підтвердження (32 біта)										
Зміщення (4 біта)	Резерв (4 біта)	CWR	ECU	URG	ACK	PSH	RST	SYN	FIN	Розмір вакна (16 біт)
Контрольна сумма заголовка (16 біт)						Вказівник даних (16 біт)				
Параметри і вирівнювання										
Дані										

– UDP-заголовок. На Рисунку 1.6 відображено формат UDP-пакету.

Рисунок 1.6 – Формат UDP-пакету

IP-заголовок	
Порт відправника (16 біт)	Порт отримувача (16 біт)
Довжина (16 біт)	Контрольна сума (16 біт)
Дані	

– ICMP-заголовок. На Рисунку 1.7 відображено формат ICMP-пакету.

Рисунок 1.7 – Формат ICMP-пакету

IP-заголовок		
Тип (8 біт)	Код (8 біт)	Контрольна сума (16 біт)
Ідентифікатор (16 біт)		Номер черги (16 біт)
Дані		

1.7 Статистичні дані

Експерти «Лабораторії Касперського» провели дослідження і виявили, що в 2018 році DDoS-атаки були здійснені на кожну шосту російську компанію. ЗА даними спеціалістів, впродовж року було здійснено близько 120 тисяч атак, які були направлені на 68 тисяч ресурсів по всьому світу. В Россії кіберзлочинці частіше за все вибирали своєю ціллю великий бізнес — у 20 % випадків, середній і малий бізнес — 17 %[11].

DDoS-атаки були націлені на створення проблем в роботі головної сторінки сайту компаній (55 % атак), виведення з ладу комунікаційних сервісів і пошти (34 %), функції, що дозволяють користувачу війти в систему (23 %). Також експерти з'ясували, що 18 % DDoS-атак зафіксовано на файлові сервіси і 12 % — на сервіси по здійсненню фінансових операцій. За результатами аналізу статистики DDoS-атак з використанням ботнетів у першому кварталі 2015 року, що проводить Kaspersky Lab, Україна ввійшла до топ-15 країн, в яких спостерігається найбільша кількість DDoS-атак.

В першому кварталі 2018 року в Україні було здійснено 122 DDoS-атаки з використанням ботнетів, що перевищує показник за аналогічний період 2019 року. Всього в світі за той період було здійснено більш ніж 23 тисячі DDoS-атак з використанням ботмереж на ресурси 76 країн. В першому кварталі ботнети атакували більш ніж 12 тисяч жертв по всьому світу. Росія займає п'яте місце в світі за кількістю DDoS-атак на її сайти. Найбільша кількість кіберзлочинів

здійснюється в Китаї, США, Кореї та Канаді. Однак атаки частіше за все здійснюються китайськими та російськими хакерами.

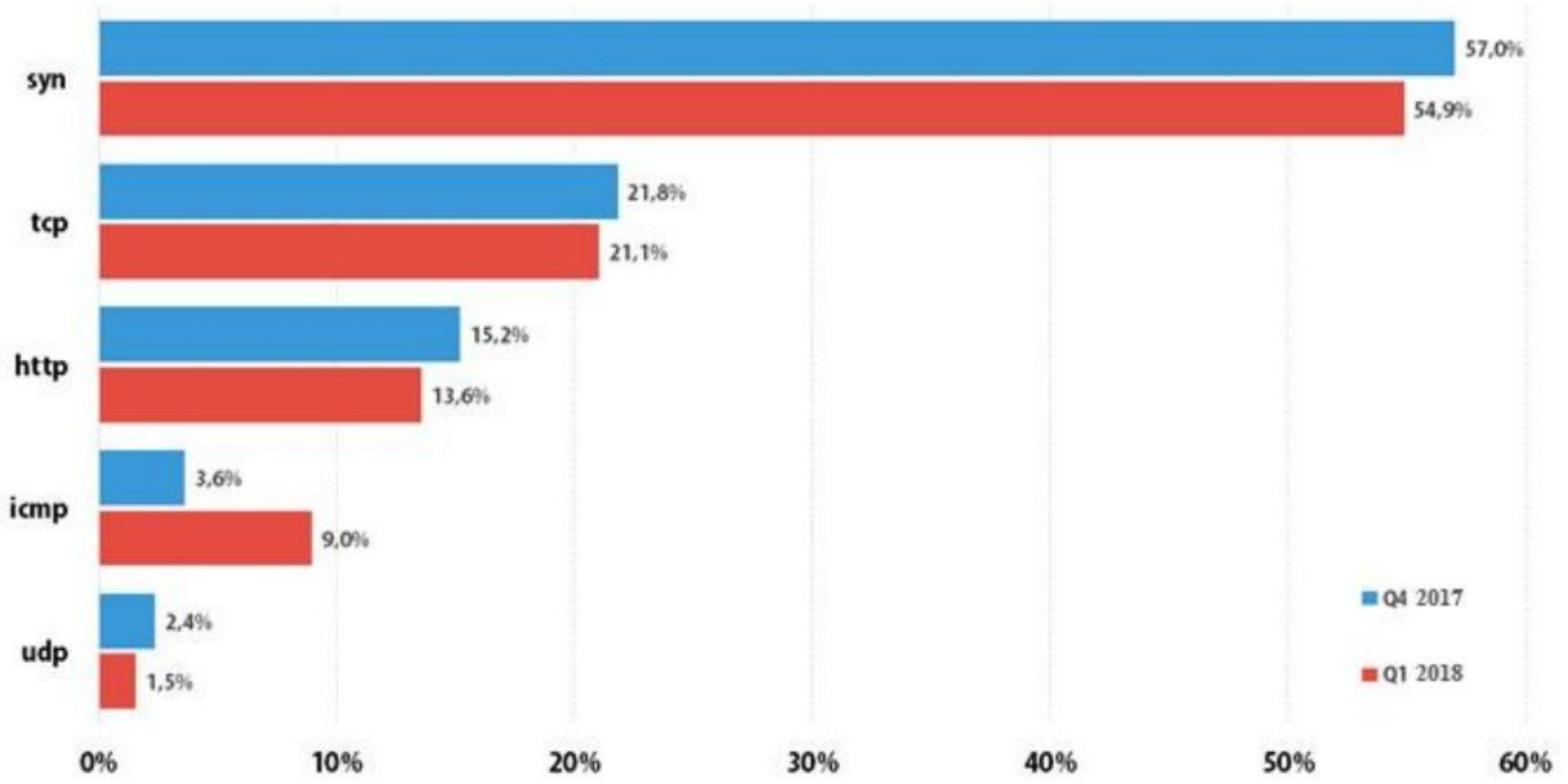


Рисунок 1.8 – Розподіл атак за типом, за 4-й квартал 2017 і 1-й квартал 2018

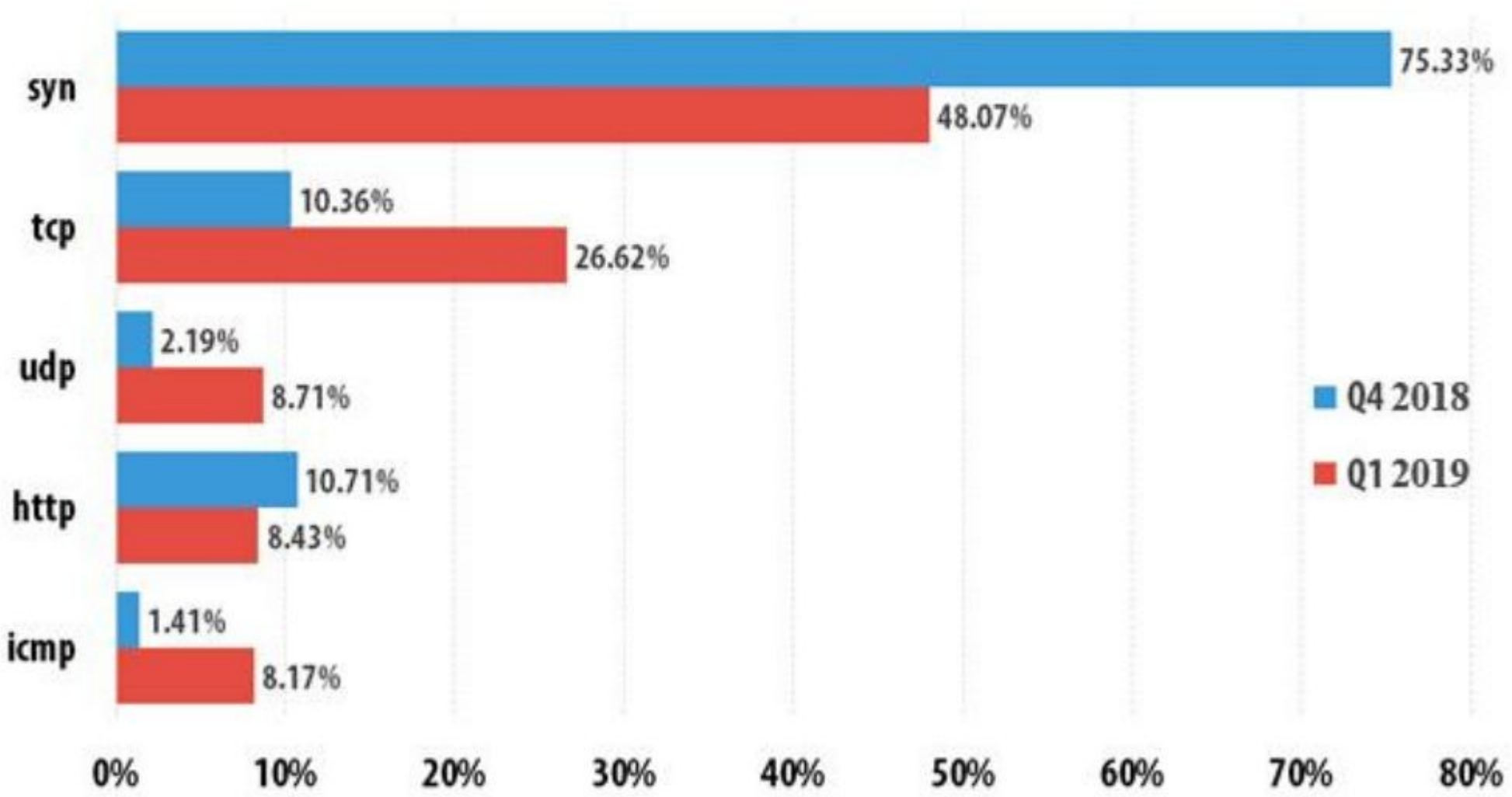


Рисунок 1.9 – Розподіл атак за типом, за 4-й квартал 2018 і 1-й квартал 2019

Дивлячись на дані графіки можна легко побачити, що за той самий період в 2018 та в 2019 роках популярність DDoS-атак значно зросла. А самий популярний тип атак взагалі на чверть підвищив свій показник.

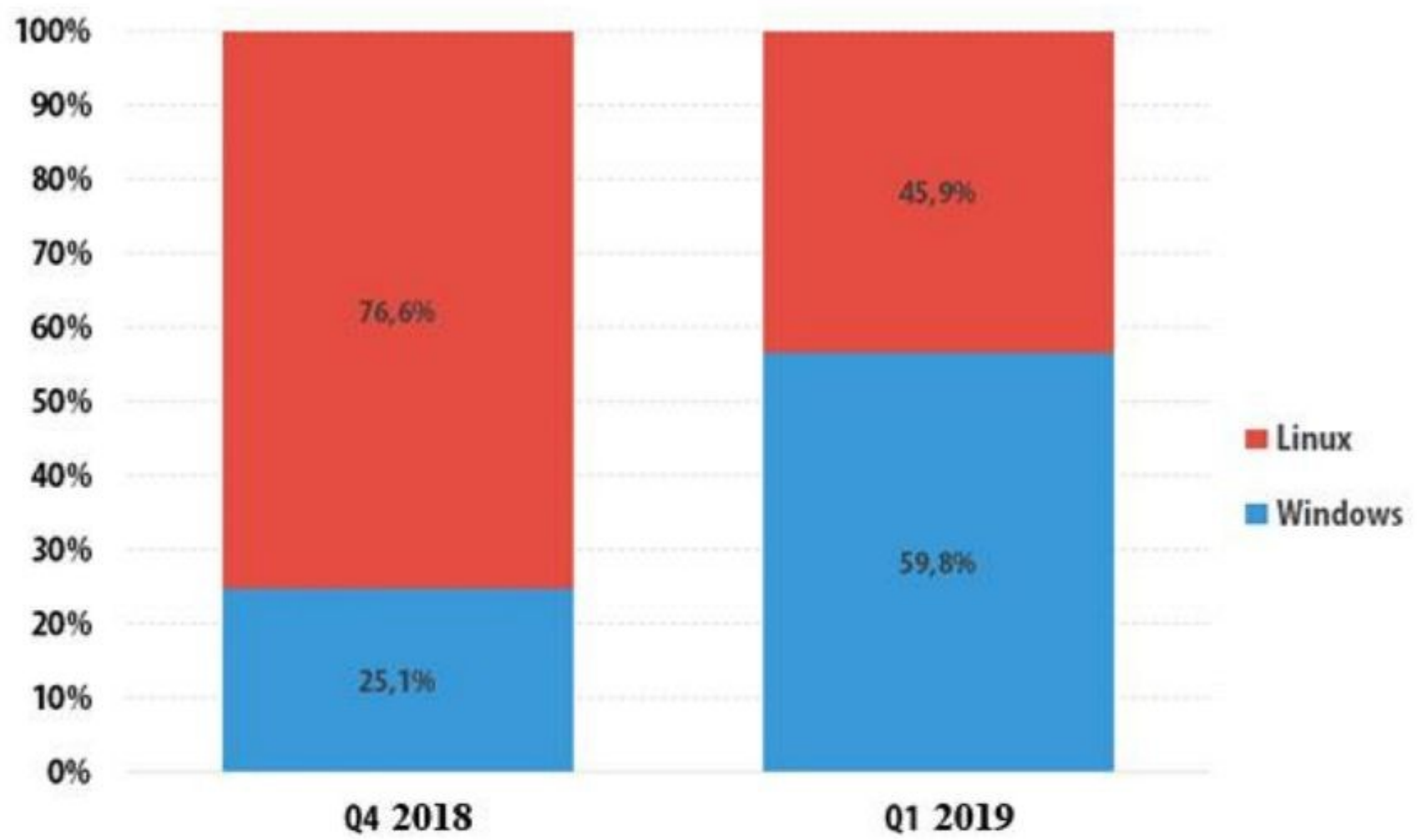


Рисунок 1.10 - Співвідношення атак з Windows- и Linux-ботнетів за 4-й квартал 2018 та 1-й квартал 2019

На графіку видно, що популярність ботнетів для Windows значно збільшилась у першому кварталі 2019 року. Це пояснюється збільшенням активності ботів сімейств Yoyo, Drive и Nitol — всі вони розроблені для Windows.

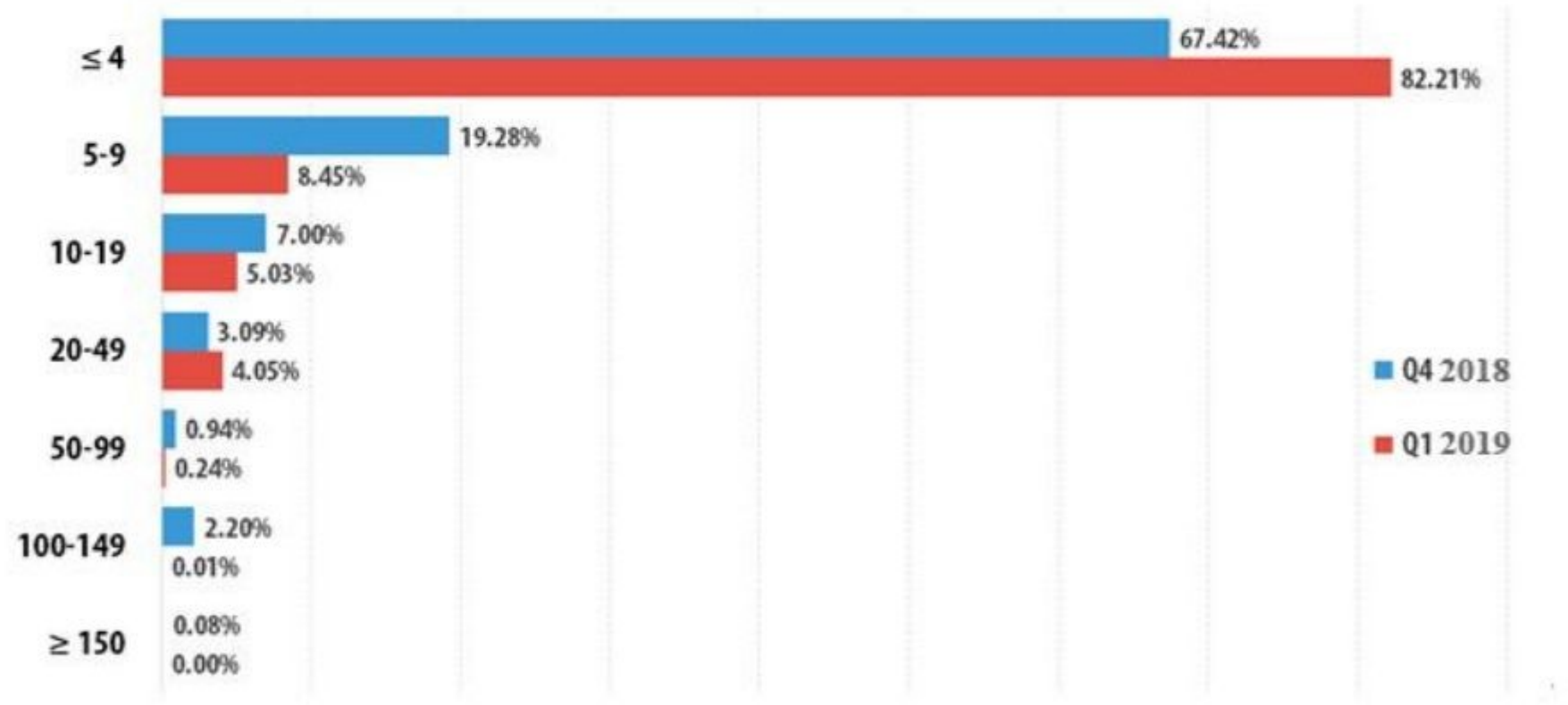


Рисунок 1.11 - Розподіл DDoS-атак за тривалістю в годинах, за 4-й квартал 2018 та 1-й квартал 2019

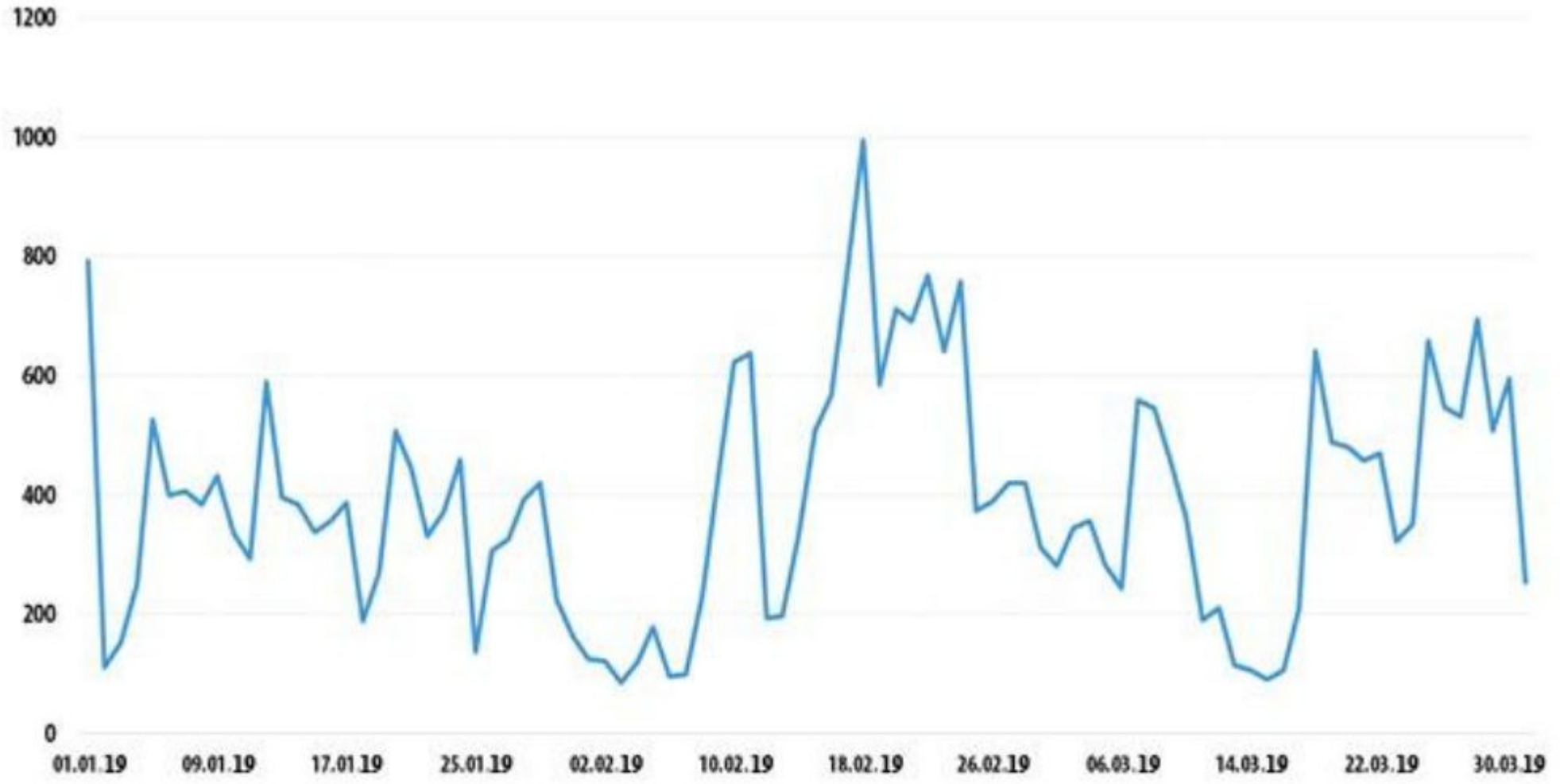


Рисунок 1.12 – Динаміка числа DDoS-атак за 1-й квартал 2019

Статистичні дані про атаки взяті з мережі інтернет та задопомогою Microsfot Excel були створені графіки, що відображають ситуацію на ринку DDoS-атак на актуальний час.

1.8 Програмні засоби для аналізу мережевого трафіку

Програмні засоби для моніторингу трафіку в мережі та відображення пакетів виникнули дуже давно. З часом дані программи прогресували і з'являлись все нові і нові функції.

1.6.1 tcpdump

tcpdump - сніфер, утиліта UNIX, що дозволяє захоплювати і аналізувати мережний трафік, що проходить через комп'ютер, на якому запущена ця програма. Програма складається з двох основних частин: частини захоплення пакетів (звернення до бібліотеки, libcap (Linux) або pcap (Windows)) і частини відображення захоплених пакетів (яка на рівні вихідного коду є модульною і для підтримки нового протоколу досить додати новий модуль)[12].

Частина захоплення пакетів (при запуску) передає "вираз вибору пакетів" (що йде після всіх параметрів командного рядка) безпосередньо бібліотеці захоплення пакетів, яка перевіряє правильність синтаксису виразу, компілює його (у внутрішній формат даних), а потім копіює у внутрішній буфер програми мережні пакети, які проходять через вибраний мережевий інтерфейс і задовольняють умовам у виразі.

Частина відображення пакетів по черзі вибирає захоплені пакети з внутрішнього буфера, і виводить їх на стандартний вивід в форматі зрозумілому людині, згідно із заданим рівнем детальності. Якщо задано докладний вивід пакетів, програма перевіряє для кожного мережевого пакету, чи є у неї модуль розшифровки даних, у разі наявності, відповідною підпрограмою витягує (і відображає) тип та інші параметри пакету в протоколі.

1.6.2 Wireshark

Wireshark — програма для аналізу мережевих пакетів Ethernet і інших мереж (сніфер) з вільним вихідним кодом. Має графічний інтерфейс користувача. У червні 2006 року проект був перейменований на Wireshark через проблеми з торговою маркою[13].

Функціональність, яку надає Wireshark, дуже схожа з можливостями програми tcpdump, проте Wireshark має графічний інтерфейс користувача і значно більше можливостей із сортування і фільтрації інформації. Програма дозволяє користувачеві переглядати весь трафік, що проходить по мережі, в режимі реального часу, переводячи мережну карту в promiscuous mode.

Wireshark — це програма, яка розпізнає структуру найрізноманітніших мережевих протоколів, і тому дозволяє розібрати мережевий пакет, відображаючи значення кожного поля протоколу будь-якого рівня. Оскільки для захоплення пакетів використовується pcap, існує можливість захоплення даних тільки з тих мереж, які підтримуються цією бібліотекою. Проте, Wireshark вміє працювати з

безліччю форматів початкових даних, відповідно, можна відкривати файли даних, захоплених іншими програмами, що розширює можливості захоплення[14].

1.6.3 Snort

Snort — вільна система виявлення та запобігання атак, котра комбінує в собі методи зіставлення по сигнатурам, засоби для інспекції протоколів і механізми для виявлення аномалій[15].

Виконує протоколювання, аналіз, пошук по вмісту, а також широко використовується для активного блокування або пасивного виявлення цілої низки нападів і зондувань, таких як спроби атак на переповнювання буферу, приховане сканування портів, атаки на веб-застосунки, SMB-зондування і спроби визначення операційної системи. Програмне забезпечення в основному використовується для запобігання проникненню, блокування атак, якщо вони мають місце.

Може працювати спільно з іншим програмним забезпеченням, наприклад, SnortSnarf, sguil, OSSIM і BASE (які забезпечують візуальне представлення даних вторгнення). З доповненнями від Bleeding Edge Threats підтримує антивірусне сканування потоків пакетів ClamAV і аналіз мережевих аномалій SPADE на мережевому і транспортному рівнях мережі, можливо, з урахуванням історії змін.

1.6.3 AutoScan – Network

AutoScan - це програма для аналізу та керування комп'ютерною мережею.

Головна мета програми - це показати список пристроїв, підключених до мережі.

Особливості програми є те, що вона дозволяє багатопотокове сканування та автоматично виявляє мережу в якій працює. Програмний засіб складає низьке завантаження для мережі та може працювати з декількома сайтами. Підмережі скануються автономно без будь-якого втручання людини. Додавання нового обладнання підключеного до мережі відбувається в реальному часі. Для перевірки його мережі не потрібна конфігурація.

Коротка порфівняльна характеристика перерахованих вище програм наведена у таблиці 1.7. На основі цієї таблиці можна зробити висновок, що на сьогоднішній день жодна з програм не націлена самае на виявлення DDoS-атак, тому є доцільною розробка програмного засобу для їх виявлення.

Таблиця 1.2 – Порівняльна характеристика ПЗ для аналізу трафіку мережі

Назва програми	Операційна система	Розмір програми	Функції
Tcpdump	Linux	3 мб	Консольний інтерфейс, аналіз пактів
Wireshark	Linux/Windows	56 мб	Графічний інтерфейс, аналіз пакетів
Snort	Кросплатформений	3.3 мб	Графічний інтерфейс, аналіз пакетів, виявлення атак
AutoScan	Linux/Windows	37 мб	Консольний інтерфейс, керування мережею

У даному розділі було проведено глибинний аналіз видів DDoS-атак, статистичний аналіз, аналіз програмних засобів для роботи з мережею та інших необхідних теоритичних компонентів для подальшої розробки методу виявлення DDoS-атак.

2 РОЗРОБКА МЕТОДУ ВИЯВЛЕННЯ DDOS-АТАК

2.1 Розробка методу виявлення DDoS-атак

Щоб мати можливість виявити різні типи DDoS-атаки, ми повинні зосередитися на пошуку загальних атрибутів цих атак. Одним з таких атрибутів є різноманітність вихідних IP-адрес, з яких надходять атакуючі пакети. Формула ентропії Шеннона може бути використана для вимірювання різноманітності пакетів. У нашому підході ентропія різноманітності пакетів H (Z) обчислюється за допомогою рівняння (1), де Zi являє собою кількість пакетів, що обмінюються між парою зв'язку (вихідною і вихідною IP-адресами, або src-dst IP) та p(Zi) представляє ймовірність появи, обчислену діленням загальної кількості пакетів на Zi.

$$H(z) = -\sum_{i=1}^{n} z_i \, log(p(z_i)) \quad (1)$$

Потім значення ентропії нормалізується з максимальною ентропією (2), де N - кількість різних пар зв'язку (src-dst IP).

$$H(N) = log(N)/log(2) \quad (2)$$

Значення ентропії може вказувати на відсоток різноманітності пакетів. З настанням DDoS-атаки різноманітність, і згодом ентропія зміниться. Чи збільшиться чи зменшиться, залежить від розміру мережі та кількість зловмисників.

Реальний трафік - це що завгодно, але не регулярна різноманітность пакетів. Різні послуги вимагають різної швидкості передачі, що створює нерегулярні структурні потоки даних. Саме тому значення ентропії можуть змінюватися сильно щосекунди. Такі зміни нам заважають встановленню порогу виявлення DDoS-атаки. Ентропія даних повинна бути відфільтрована таким чином, що для прийняття рішень використовуються лише орієнтовні значення. Метод пропонує фільтрувати, наприклад, групуючи дані ентропії в більші інтервали по 10с, і

приймаючи найвище значення ентропії як репрезентативне значення для кожної групи (Рис. 2.1).

$$F(z) = max(zi, i = 1..10) \quad (3)$$

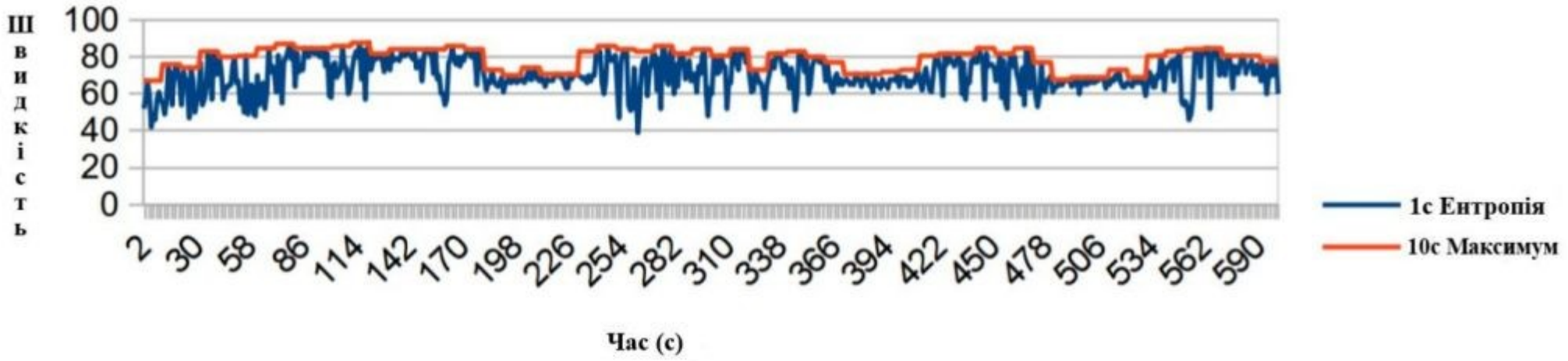


Рисунок 2.1 – Найвищі значення швидкості у проміжках в 10с

Тим не менш, через падіння значення ентропії, тривога може спрацьовувати у випадку коротких комунікацій із декількома з'єднаннями. Це стосується веб-сторінки з великою кількістю зовнішніх джерел, наприклад, розповсюдження поштового сервера на величезну кількість адрес (наприклад, список розсилки). Ми вирішуємо цей випадок шляхом подальшого уповільнення виявлення, і для цього використовуємо алгоритм (4) експоненціальної ковзної середньої (ЕКС).

$$\Delta_n^N = (1 - \alpha)^{N-1} x_{n-N-1} + \alpha \sum_{s=0}^{N-2} (1-a)^s x_{n-s} \quad (4)$$

$$\text{де } a = \frac{2}{N+1}$$

Значення ЕКС розраховуються на відфільтрованих значеннях ентропії для отримання середньої ентропії за певний період. Ця методика успішно долає ситуації з короткими сплесками декількох з'єднань, але все ж є проблеми з більш тривалими пакетами, які не є DDoS-атаками. Для подальшого вирішення питання пакетів і встановлення процедури прийняття рішень для автоматизованого виявлення DDoS пропонується використовувати два індикатори ЕКС для виявлення. Один показник ЕКС відфільтрованих значень ентропії використовує для коротких періодів, наприклад, дві вибірки. Це швидка змінна ЕКС (ШЕКС). Інший показник ЕКС використовується для тривалих періодів, наприклад, шість зразків, що являє собою повільну змінну ЕКС (ПЕКС). Два значення показують швидкі та

повільні тенденції зміни ентропії відповідно (рис. 2.2). Обчислюючи різницю між ШЕКС та ПЕКС (5), ми можемо отримати чітку вказівку на тенденцію до зростання чи спадання фільтрованих значень ентропії.

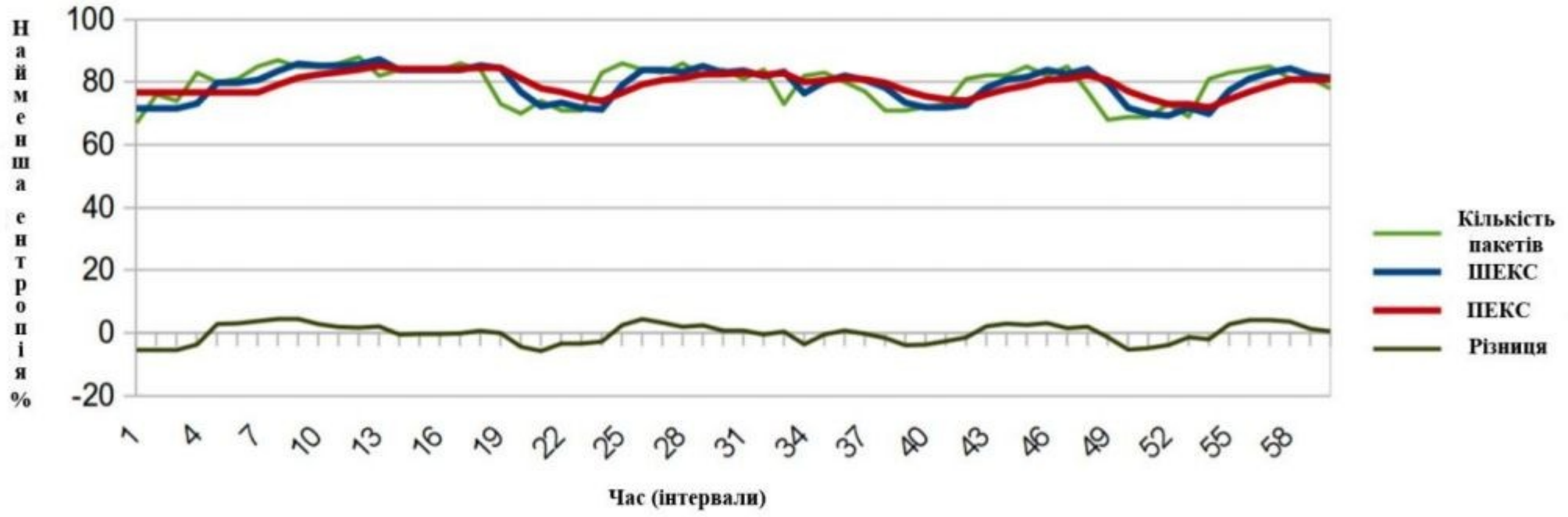


Рисунок 2.2 – Значення ШЕКС, ПЕКС та їх різниця у часі

У нашій тестовій мережі, тенденція до зменшення ентропії може означати початок DDoS-атаки, оскільки різноманітність пакетів з часом збільшується (10s фільтрація * 2 вибірки = 20s для реакції ШЕКС). З іншого боку, зростаюча тенденція може свідчити про закінчення атаки DDoS.

$$dif\ f(x) = \text{ШЕКС}(x) - \text{ПЕКС}(x) \quad (5)$$

Для тонкої настройки алгоритму прийняття рішень ми включили пороги різниці ЕКС. Наприклад, сигнал тривоги активується лише в тому випадку, якщо diff(x) менше -1,2, а деактивується лише в тому випадку, якщо він перевищує 2. Така оптимізація виявлення можлива в мережах із певним профілем трафіку. Запропонована процедура може бути оптимізована для виявлення великого спектру DDoS-атак. Але виявлення ентропії базується лише на різноманітності пакетів. У певних випадках корисність такого детектора обмежена. Отже, ми не можемо базувати свій метод лише на одному загальному атрибуті DDoS-атак. Окрім різноманітності пакетів, ми вирішили використовувати швидкість пакетів як ще один загальний атрибут атак.

Використовуючи абсолютно ту саму процедуру, що описана для ентропійних значень, ми можемо обчислити різницю швидкостей пакетів ЕКС. З періодом від

однієї секунди та збільшенням таймеру до 10 секунд вибирається найвище значення. На основі цих вибірок ШЕКС розраховується за допомогою чотирьох вибірок, які мають значення швидкості пакетів з більшою дисперсією, ніж ентропія. ПЕКС використовує інтервал у вісім зразків. Потім обчислюється різниця між ШЕКС та ПЕКС, і застосовується поріг для тонкого налаштування виявлення (рис. 2.3).

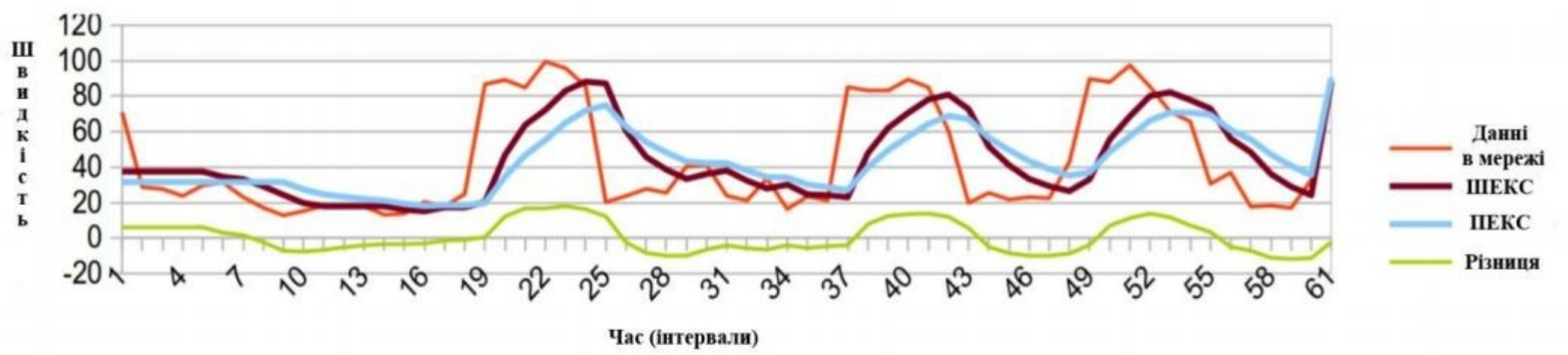


Рисунок 2.3 – ШЕКС, ПЕКС та значення їх різниці для швидкості пакетів

Нарешті, наш механізм виявлення складається із застосування порогу як до ентропії різниці ЕКС, так і до різниці швидкості ЕКС. На рис. 2.4 представлений повний алгоритм з параметрами, описаними в Таблиці 2.1. Для більш легкої ідентифікації запропонований спосіб відтепер у тексті називається 4ЕКС.

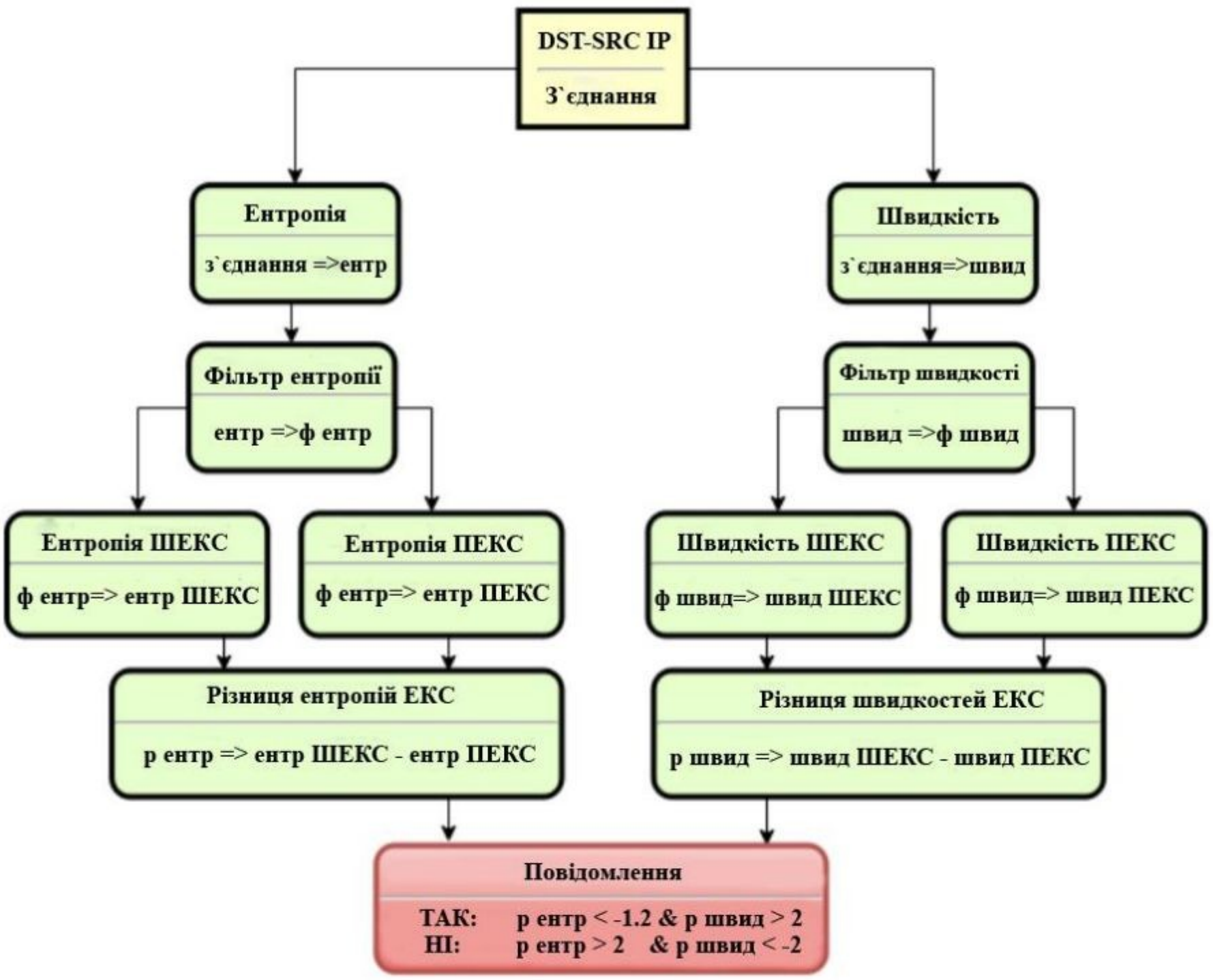


Рисунок 2.4 – Алгоритм роботи 4ЕКС

Таблиця 2.1 – Параметри 4ЕКС

Параметр	Значення	Опис
eksFastInterval	2 приклади (20 секунд)	ШЕКС інтервал для даних ентропії
eksSlowInterval	4-6 прикладів	ПЕКС інтервал для даних ентропії
xEntAlarm	-0.74	Порогове значення різниці ентропій
xEntNoAlarm	0.10	Порогове значення різниці ентропій, що вказує на закінчення атаки
eksPacketFastInterval	4 приклади	ШЕКС для даних про швидкість передачі пакетів
eksPacketSlowInterval	8 прикладів	ПЕКС для даних про

		швидкість передачі пакетів
xPktAlarm	0.10	Порогове значення різниці швидкостей ЕКС
xPktNoAlarm	-0.50	Порогове значення різниці швидкостей ЕКС, що вказує на закінчення атаки

На основі методу розробленого вище було прийнято рішення розробити програмний засіб для практичної реалізації даного методу виявлення DDoS-атак.

2.2 Технологія Netfilter

Netfilter - це платформа, надана Linux, яка дозволяє виконувати різні операції, пов'язані з мережею, у формі індивідуальних обробників. Netfilter пропонує різні функції та операції з фільтрації пакетів, передачі мережевої адреси та порту, які забезпечують функціональні можливості, необхідні для керування пакетами через мережу, а також для забезпечення заборони доступу пакетів до чутливих місць розташованих в комп'ютерній мережі.

Netfilter являє собою набір дій всередині ядра Linux, що дозволяє конкретним модулям ядра пов’язувати функції зворотного виклику з мережевим стеком ядра. Ці функції, як правило, застосовуються до трафіку у вигляді правил фільтрування та модифікації, викликаються для кожного пакета, який відповідає певному стану у мережевому стеку [16].

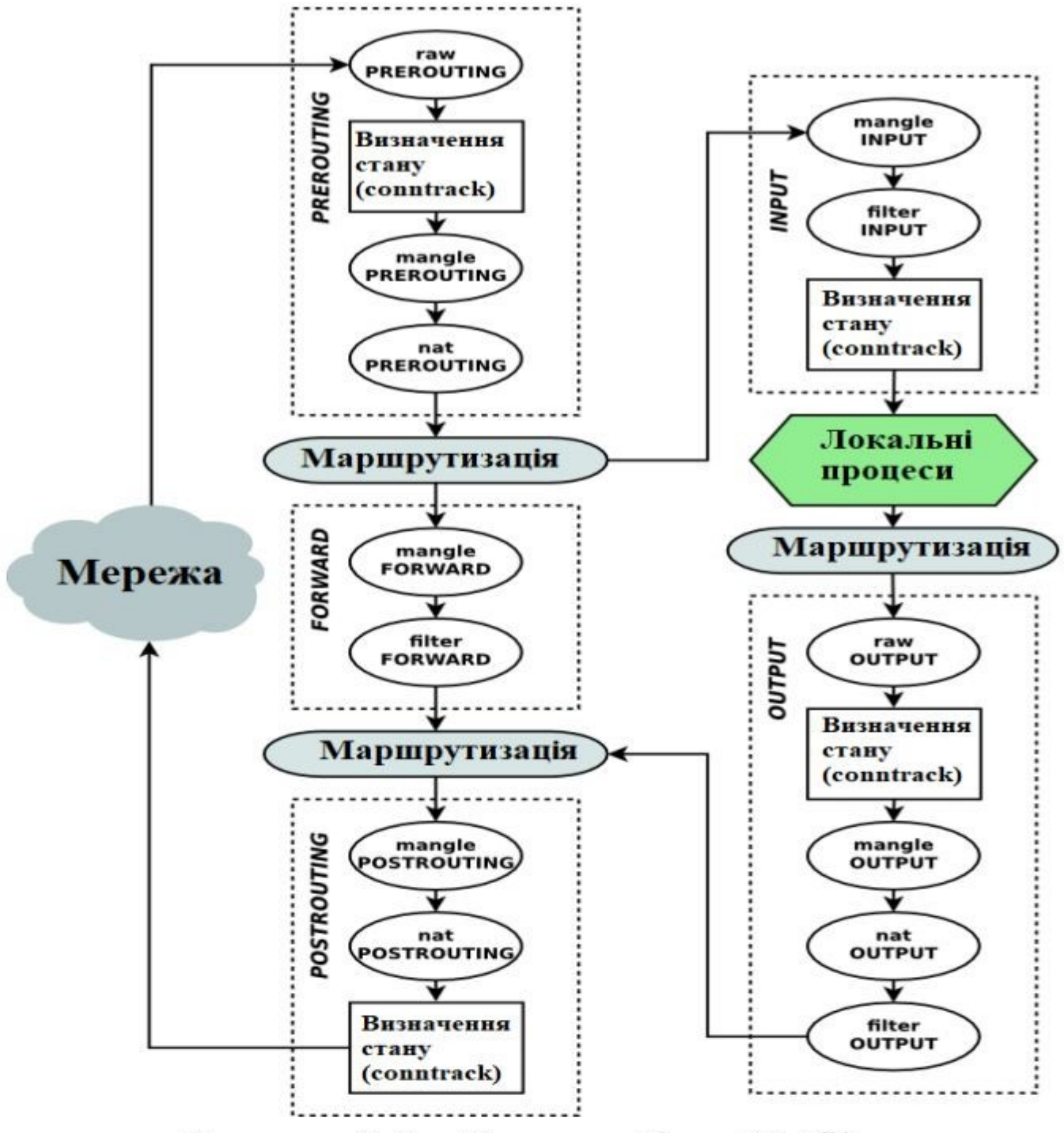


Рисунок 2.5 – Схема роботи Netfilter

В системі netfilter пакети пропускаються через ланцюги. Ланцюжок є упорядкованим списком правил, і кожне правило може містити критерії та дію або перехід. Коли пакет проходить через ланцюг, система netfilter по черзі перевіряє, чи відповідає пакет всім критеріям чергового правила, і якщо так, то виконує дію (якщо критерії в правилі немає, то дія виконується для всіх пакетів, що проходять через правило). Варіантів можливих критеріїв дуже багато. Наприклад, пакет відповідає критерію - джерело 192.168.1.1, якщо в заголовку пакета вказано, що відправник - 192.168.1.1. Самий простий тип переходу, --jump, просто пересилає пакет в початок іншого ланцюжка. Також при допомозі --jump можна вказати дію. Стандартні дії, доступні в усіх ланцюжках - ACCEPT (пропустити), DROP (видалити), QUEUE (передавати на аналіз зовнішню программу), і RETURN (повернути на аналіз у попередній ланцюжок).

Ланцюжок Відповідальностей забезпечує обробку об’єкта, шляхом передачі його по ланцюжку доти, доки не буде здійснена обробка якоюсь із ланок.

Ланцюжок Відповідальностей призначений для уникнення зв'язності відправника запиту із його адресатом, шляхом надання іншим об'єктам можливість обробити запит, він передає отримані об'єкти вздовж ланцюжка допоки якась ланка не обробить об'єкт[17]. На рисунку 2.6 зображено схему роботи Ланцюжка Відповідальностей.

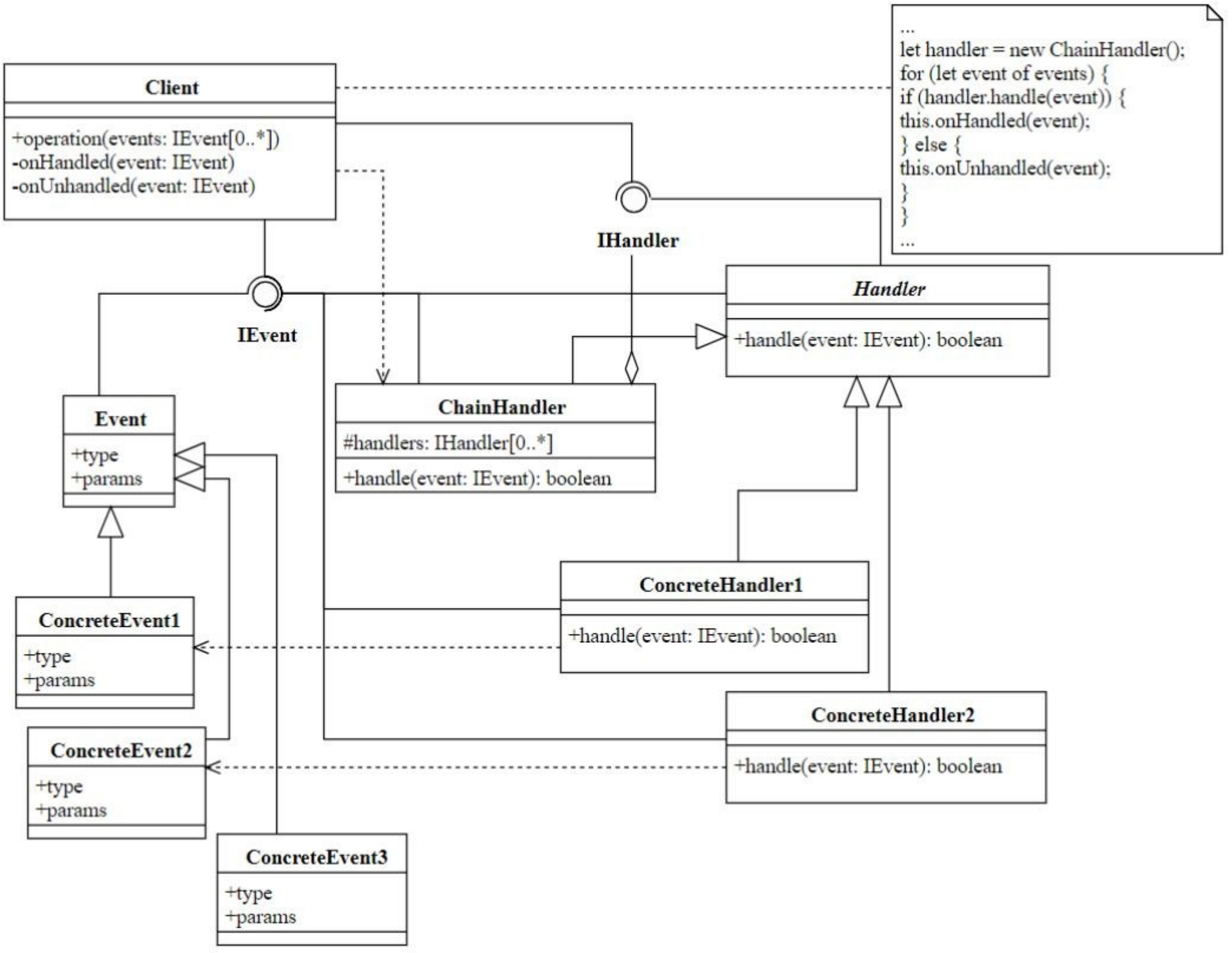


Рисунок 2.6 – Схема роботи Ланцюжка Відповідальностей

2.3 Бібліотека Pcap

Бібліотека Pcap дозволяє створювати програми аналізу мережевих даних, що надходять на мережеву карту комп'ютера. Прикладом програмного забезпечення, що використовує бібліотеку Pcap, служить програма Wireshark. Різноманітні програми моніторингу та тестування мережі, сніфери використовують цю бібліотеку. Вона призначена для використання спільно з мовами C / C ++, а для роботи з бібліотекою на інших мовах, таких як Java, .NET, використовують

обгортки. Для Unix-подібних систем це бібліотека libpcap, а для Microsoft Windows - WinPcap. Програмне забезпечення мережевого моніторингу може використовувати libpcap або WinPcap, щоб захопити пакети, які подорожують по мережі, і (в новіших версіях) для передачі пакетів в мережі. Програми, написані на основі libpcap або WinPcap, можуть захопити мережевий трафік, аналізувати його. Файл захопленого трафіку зберігається в форматі, зрозумілому для додатків, що використовують Pcap[18].

libpcap та WinPcap забезпечують механізми захоплення та фільтрації пакетів багатьох відкритих і комерційних мережевих інструментів, включаючи аналізатори протоколів (сніфери пакетів), мережеві монітори, системи виявлення вторгнення в мережу, генератори трафіку та мережеві тестери.

libpcap та WinPcap також підтримують збереження захоплених пакетів у файлі та читання файлів, що містять збережені пакети; додатки можуть бути написані за допомогою libpcap або WinPcap, щоб мати можливість фіксувати мережевий трафік та аналізувати його, або читати збережений знімок і аналізувати його, використовуючи той самий код аналізу. Файл захоплення, збережений у форматі, який використовує libpcap та WinPcap, можна читати програмами, що розуміють цей формат.

2.4 Обґрунтування вибору програмного засобу

Реалізація поставленої задачі, а саме розробка програмного засобу для виявлення DDoS-атак буде проводитися за допомогою мови програмування JavaScript, а саме за допомогою Node.js, в програмному середовищі Visual Studio Code.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Node.js призначений для відокремленого виконання високопродуктивних мережевих застосунків на мові JavaScript. Функції платформи не обмежені створенням серверних скриптів для веб, платформа може використовуватися і для створення звичайних клієнтських і серверних мережевих програм. Для забезпечення виконання JavaScript-коду використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів. Всі системні виклики, що спричиняють блокування, виконуються всередині пула потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали. Проаналізувавши основні особливості мови програмування JavaScript, сформу-льовано найбільш помітні переваги досліджуваної мови програмування.

Оскільки дуже важливу роль відіграють розмір, швидкість виконання, наявність дружнього графічного інтерфейсу буде доцільно обрати саме цю мову програмування для реалізації поставленої задачі.

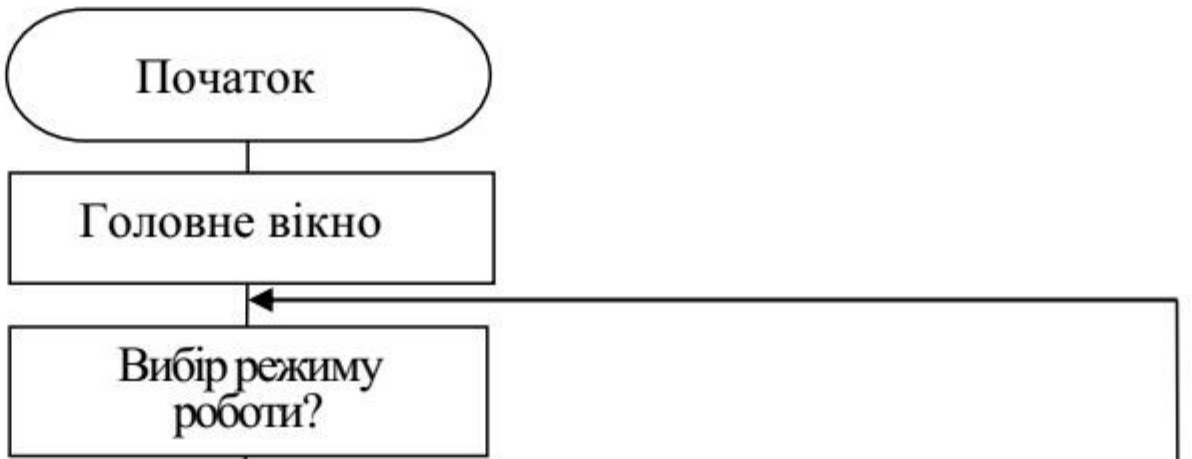
2.5 Розробка загальної схеми роботи програми

Програмний засіб розрахований на користувача, який вміє хоча б мінімально працювати в консолі. Даний засіб використовує повністю консольний інтерфейс, тому всі дії в програмі виконуються за допомогою консольних команд.

На основному екрані можна за допомогою команд керування виконувати певні дії, які дозволяють керувати роботою програми, а саме:

– передивлятись інформацію про автора;
– читати інструкції по роботі з програмою;
– виконувати налаштування;
– або безпосередньо перейти до процесу виявлення DDoS-атак.

Загальна схема роботи програми наведена на рисунку 2.7.



Вихід Пошук атак Інформація про розробника Інструкція Налаштування

Рисунок 2.7 – Схема функціонування програми

Отже, із загальної схеми стає зрозуміло, як працює програмний засіб та які режими роботи є.

2.6 Розробка програмного засобу

Усі теоритичні напрацювання, що були виконані при розробці методу були перенесені у програмний засіб, для практичного використання даного методу.

Нижче відображено частину коду, що показує частковий алгоритм роботи програми, що відповідає за вивлення ICMP-flood атаки.

```
public handle(this: ICMPFloodFilter, linkType: LINK_TYPE, packet: EthernetPacket | NullPacket | RadioPacket | SLLPacket): boolean {
        if (!(packet instanceof RadioPacket) && packet.payload) {
            const networkLayer = packet.payload;
            if ((networkLayer instanceof IPv4 || networkLayer instanceof IPv6) && networkLayer.saddr && networkLayer.payload instanceof ICMP) {
                const transportLayer = networkLayer.payload;
                if (transportLayer.type === 8 && transportLayer.code === 0) {
                    const link = `${networkLayer.saddr} => ${networkLayer.daddr}`;
                    if (this._counter.add(link, 1)) {
                        this.alert({
```

```
                message: 'ICMP flood',
                source: link,
            });
```

Нижче відображено частину коду, що показує частковий алгоритм роботи програми, що відповідає за вивлення HTTP-flood атаки.

```
    public handle(this: Http404Filter, linkType: LINK_TYPE, packet: EthernetPacket | NullPacket | RadioPacket | SLLPacket): boolean {
        if (!(packet instanceof RadioPacket) && packet.payload) {
            const networkLayer = packet.payload;
            if ((networkLayer instanceof IPv4 || networkLayer instanceof IPv6) && networkLayer.saddr && networkLayer.payload instanceof TCP) {
                const transportLayer = networkLayer.payload;
                if (this._ports.has(transportLayer.sport || 0) && transportLayer.dataLength && transportLayer.data) {
                    const link = `${networkLayer.daddr}`;
                    if (/^HTTP[/]\d[.]\d\s+404/.test(transportLayer.data.toString()) && this._counter.add(link, 1)) {
                        this.alert({
                            message: 'HTTP/404 flood',
                            source: link,
                        });
                    }
                }
            }
        } return false;
```

Нижче відображено частину коду, що показує частковий алгоритм роботи програми, що відповідає за вивлення SYN-flood атаки.

```
    public handle(this: SynFloodFilter, linkType: LINK_TYPE, packet: EthernetPacket | NullPacket | RadioPacket | SLLPacket): boolean {
        if (!(packet instanceof RadioPacket) && packet.payload) {
            const networkLayer = packet.payload;
            if ((networkLayer instanceof IPv4 || networkLayer instanceof IPv6) && networkLayer.saddr && networkLayer.payload instanceof TCP) {
                const transportLayer = networkLayer.payload;
                if (transportLayer.flags && transportLayer.flags.syn) {
                    const link = `${networkLayer.saddr}`;
                    if (this._counter.add(link, 1)) {
                        this.alert({
```

```
                message: 'SYN flood',
                source: link,
            });
        }
        return true;
```

Дані атаки були обрані через свою частоту використання на основі визначених у першому розділі статистичних даних.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Тестовий сценарій

Моделювання комп'ютерних мереж відтворює специфічні сценарії роботи мережі. Зазвичай ці сценарії базуються на конкретному випадку і оцінка, таким чином, обмежується цим конкретним випадком. Найважливішим завданням для будь-якого методу є тестування в реальній ситуації, коли трафік є більш непередбачуваним. Розроблений метод був перевірений у реальному комп'ютерному мережевому середовищі (рис. 5). Для тестування було використоно робочу комп'ютерну мережу, яка включала близько 300 співробітників фірми, керівників та ін. особового складу, та близько 20 серверів. З метою тестування було розробилено прототип системи виявлення вторгнень (СВВ), здатну збирати трафік, що надходить з мережі, обчислювати змінні, визначені запропонованим методом, приймати рішення тривоги та зберігати змінні в базі даних для подальшого аналізу. Прототип СВВ розміщується на одному з серверів.

Після вилучення даних заголовка IP для кожної пари зв'язку (src-dst IP), кількість пакетів підраховується, поки не закінчиться інтервал вибірки. Інформація про протоколи та прапорці TCP також зберігається.

Наприкінці кожного інтервалу вибірки збираються дані про пари зв’язку, зберігаються в базі даних, лічильник скидається і застосовується запропонований метод виявлення.

У рамках сценарію тестування було створено: базовий рівень, де збираються дані з мережі, коли атаки немає. Аналіз трафіку підтверджує, що значення ентропії у разі нормальної роботи (без активної атаки) не є стабільними, що виключає можливість використання простих порогових значень.

Для перевірки працездатності запропонованого методу було встановлено ботнет-мережу всередині робочої мережі (рис.3.1). Використовуючи службу Microsoft Windows Active Directory розповсюдсюджено файл сценарію атаки на деякі комп’ютери в офісах. Інтенсивність проведеної атаки вибиралася ретельно, щоб не зашкодити нормальній роботі офісної мережі.

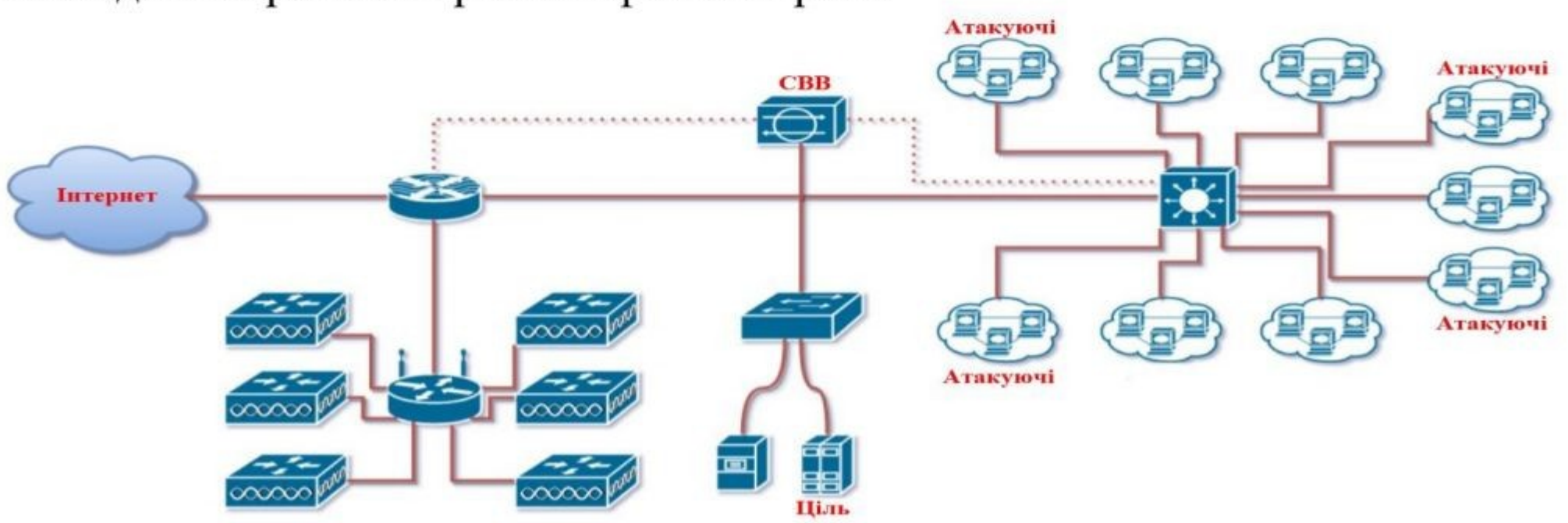


Рисунок 3.1 – Вигляд тестової мережі

Для тестування наступних атак був розроблений файл сценарію: ICMP flood атака з високою швидкістю пакетів на вказану ціль. Пакети відправляються з максимальною швидкістю. TCP SYN-атака, яка надсилає налаштовану кількість пакетів SYN в секунду на задані цілі. Ця атака реалізується шляхом надсилання великої кількості пакетів SYN до цілі атаки. Цільовий сервер відповідає, надсилаючи SYN ACK пакети, але зловмисники не завершують тристороннє рукостискання TCP, тобто не надсилають пакети ACK. Таким чином, ресурси, виділені для встановлення з'єднання, не звільняються до часу очікування.

У той час як ICMP flood атака спрямована на виснаження мережевого ресурсу жертв і є представником атаки, що інтенсивно працює на мережевих ресурсах, TCP SYN є представником атаки середньої інтенсивності мережі, яка спрямована на інші типи ресурсів (наприклад, процесор, оперативна пам'ять тощо).

У разі SYN flood, зловмисник намагається вичерпати так звані відставання напіввідкритих з'єднань, пов'язаних з номером порту. Блокування - це системне обмеження на кількість структур TCP Control Block (TCB), які можуть бути резидентними в будь-який час.

Сценарій, який поширюється на комп'ютери, залишається резидентом у пам'яті після ввімкнення комп`ютора, і він відстежує конкретну цільову IP-адресу. Коли цільова IP-адреса стає доступною (тобто вона відповідає на повідомлення ICMP), всі активні комп’ютери в нашому ботнеті починають атаку на ціль. Якщо ця IP-адреса недоступна, то перевірка повторюється кожні десять секунд.
Використовуючи такий підхід, моделюється реальний ботнет.

Зазвичай майстер ботнету зберігає інструкції у фіксованому місці, а боти періодично перевіряють наявність цих інструкцій. Коли знайдеться інструкція, атака починається. Так, як і в реальній ситуації, атака починається не з усіх нападників одночасно, а натомість з нападниками, ініційованими після періоду випадкової затримки (максимум 10 секунд у нашому прикладі).

Відслідковуючи умови мережі та враховуючи графік роботи мережі, ми обрали момент для початку атаки. У цей момент IP-адреса, яку моніторив скрипт ботнету, присвоюється одному з фіктивних комп'ютерів.

Протягом інтервалу 10с комп'ютери ботнету починають безперервні атаки, які тривають до тих пір, поки ця IP-адреса присутня в мережі. Не всі комп'ютери є частиною ботнету. Active Directory використовується для розповсюдження сценарію лише на деяких комп’ютерах в працівників офісу, а сценарії виконують лише комп’ютери, за якими працюють люди. Це створює дуже мінливе середовище, коли під час тестування комп'ютери вмикаються або вимикаються, а працівники входять і виходять, і це призводить до нападу з непослідовною інтенсивністю, як у реальній DDoS-атаці.

Під час тестування на виявлення атаки TCP SYN ми можемо контролювати кількість SYN-пакетів, надісланих зловмисником, що дозволяє протестувати метод у різних випадках інтенсивності. Перший раунд TCP SYN тести ініціюються зі швидкістю атаки 100 пакетів в секунду для одного атакуючого.
Таблиця 3.1 – Характеристики SYN-flood та ICMP-flood атаки

	Високоамплітудна атака	Низькоамплітудна атака
Кількість пакетів від атакуючого в секунду	1000 pps	20 pps
Кількість атакуючих протягом атаки	33-42	55-60
Загальна швидкість передачі трафіку	6-70 kbps	3.6-29mbps pps
Трафік згенерований протягом атаки	47-390 kpps	1000-1200 kpps
Методи атаки	ICMP, UDP Flood	TCP Syn, ICMP, UDP Flood
Відсоток виявлення	100%	100%
Помилки виявлення	0%	0%

3.2 Проведення тестових DDoS-атак для випробування засобу

Перевірка алгоритму виявлення DDoS-атак проводитиметься в 4 режимах.

1) переверка на виявлення ICMP-flood атак;
2) переверка на виявлення SYN-flood атак;
3) переверка на виявлення HTTP-flood атак;
4) режим виявлення всіх вище визначених атак;

Для користування прогрмою в повному обсязі було створену функцію Help, яка показує всі можливі користувацькі команди та формат їх введення. На рис. 3.2 відображено роботу даної функції.

```
kol@kol net.alert   0 $ sudo node distribution/index.js --help
usage: index.js [-h] [-v] [-t [TIMEOUT]] -I INTERFACE [INTERFACE ...]
                [--icmp-limit [TIMEOUT:LIMIT [TIMEOUT:LIMIT ...]]]
                [--syn-limit [TIMEOUT:LIMIT [TIMEOUT:LIMIT ...]]]
                [--http404-limit [TIMEOUT:LIMIT [TIMEOUT:LIMIT ...]]]
                [--http404-port [PORT [PORT ...]]]


Network alert

Optional arguments:
  -h, --help            Show this help message and exit.
  -v, --version         Show program's version number and exit.
  -t [TIMEOUT], --timeout [TIMEOUT]
                        Session timeout
  -I INTERFACE [INTERFACE ...], --interface INTERFACE [INTERFACE ...]
                        Network interface
  --icmp-limit [TIMEOUT:LIMIT [TIMEOUT:LIMIT ...]]
                        ICMP limit
  --syn-limit [TIMEOUT:LIMIT [TIMEOUT:LIMIT ...]]
                        SYN limit
  --http404-limit [TIMEOUT:LIMIT [TIMEOUT:LIMIT ...]]
                        HTTP/404 limit
  --http404-port [PORT [PORT ...]]
                        HTTP server port
```

Рисунок 3.2 – Результат роботи функції Help

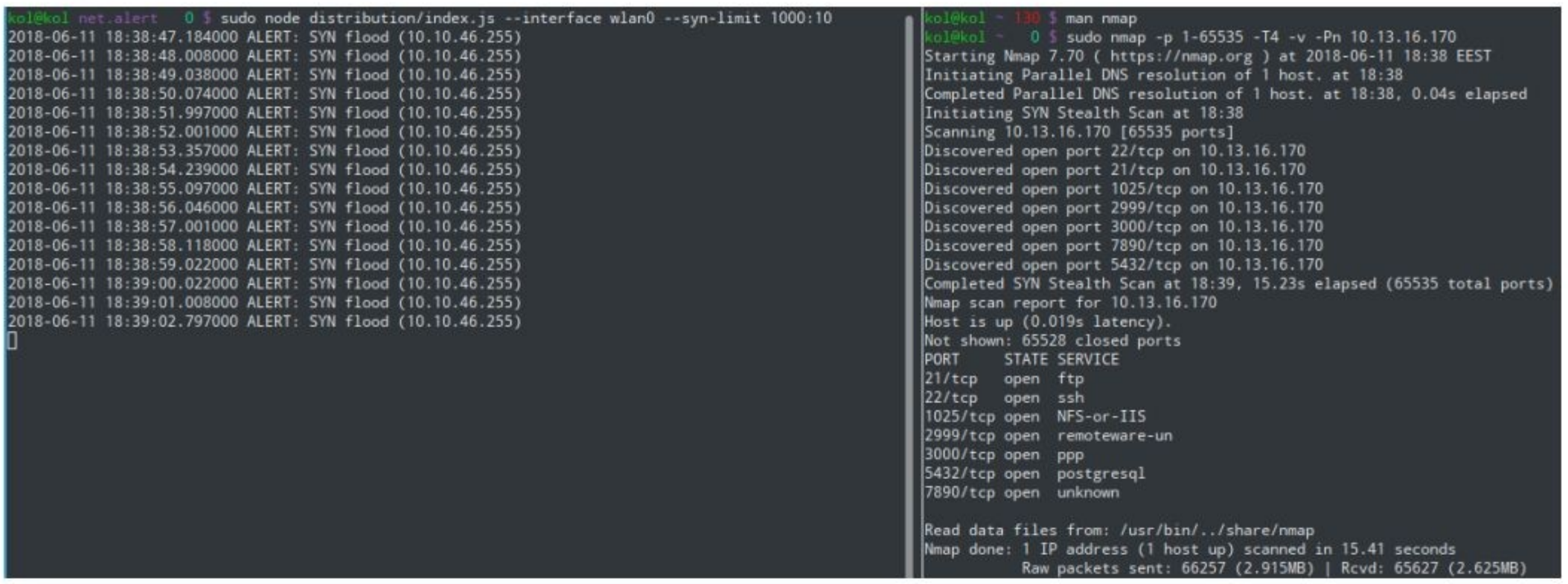
Для тестування даного програмного засобу проводились всі вище зазначені випробувальні DDoS-атаки різних видів у тестовому режимі. Як видно нижче, всі перераховані атаки були виявлені. Видно, що програмний засіб виявляє тип DDoS-атаки та ip-адресу комп'ютера з якого проводяться атаки.

```
kol@kol net.alert   0 $ sudo node distribution/index.js --interface wlan0 --icmp-limit 1000:10
2018-06-11 18:35:54.479000 ALERT: ICMP flood (10.10.46.255 => 10.13.16.170)
2018-06-11 18:35:55.491000 ALERT: ICMP flood (10.10.46.255 => 10.13.16.170)
2018-06-11 18:35:56.504000 ALERT: ICMP flood (10.10.46.255 => 10.13.16.170)
2018-06-11 18:35:57.518000 ALERT: ICMP flood (10.10.46.255 => 10.13.16.170)
```

```
kol@kol    0 $ ping -A 10.13.16.170
ping: socket: Відмовлено у доступі, attempting raw socket...
ping: socket: Відмовлено у доступі, attempting raw socket...
PING 10.13.16.170 (10.13.16.170) 56(84) bytes of data.
64 bytes from 10.13.16.170: icmp_seq=1 ttl=60 time=9.96 ms
64 bytes from 10.13.16.170: icmp_seq=2 ttl=60 time=9.96 ms
64 bytes from 10.13.16.170: icmp_seq=3 ttl=60 time=9.60 ms
no answer yet for icmp_seq=4
64 bytes from 10.13.16.170: icmp_seq=4 ttl=60 time=14.6 ms
no answer yet for icmp_seq=5
64 bytes from 10.13.16.170: icmp_seq=5 ttl=60 time=16.5 ms
no answer yet for icmp_seq=6
64 bytes from 10.13.16.170: icmp_seq=6 ttl=60 time=13.7 ms
64 bytes from 10.13.16.170: icmp_seq=7 ttl=60 time=12.4 ms
64 bytes from 10.13.16.170: icmp_seq=8 ttl=60 time=10.5 ms
64 bytes from 10.13.16.170: icmp_seq=9 ttl=60 time=10.4 ms
no answer yet for icmp_seq=10
64 bytes from 10.13.16.170: icmp_seq=10 ttl=60 time=12.2 ms
no answer yet for icmp_seq=11
64 bytes from 10.13.16.170: icmp_seq=11 ttl=60 time=13.4 ms
64 bytes from 10.13.16.170: icmp_seq=12 ttl=60 time=11.5 ms
64 bytes from 10.13.16.170: icmp_seq=13 ttl=60 time=11.4 ms
64 bytes from 10.13.16.170: icmp_seq=14 ttl=60 time=11.2 ms
64 bytes from 10.13.16.170: icmp_seq=15 ttl=60 time=11.1 ms
64 bytes from 10.13.16.170: icmp_seq=16 ttl=60 time=11.0 ms
64 bytes from 10.13.16.170: icmp_seq=17 ttl=60 time=11.1 ms
64 bytes from 10.13.16.170: icmp_seq=18 ttl=60 time=10.5 ms
no answer yet for icmp_seq=19
64 bytes from 10.13.16.170: icmp_seq=19 ttl=60 time=13.0 ms
64 bytes from 10.13.16.170: icmp_seq=20 ttl=60 time=11.0 ms
no answer yet for icmp_seq=21
64 bytes from 10.13.16.170: icmp_seq=21 ttl=60 time=11.5 ms
64 bytes from 10.13.16.170: icmp_seq=22 ttl=60 time=11.1 ms
no answer yet for icmp_seq=23
64 bytes from 10.13.16.170: icmp_seq=23 ttl=60 time=11.4 ms
64 bytes from 10.13.16.170: icmp_seq=24 ttl=60 time=10.8 ms
64 bytes from 10.13.16.170: icmp_seq=25 ttl=60 time=10.7 ms
64 bytes from 10.13.16.170: icmp_seq=26 ttl=60 time=10.6 ms
64 bytes from 10.13.16.170: icmp_seq=27 ttl=60 time=10.5 ms
64 bytes from 10.13.16.170: icmp_seq=28 ttl=60 time=10.8 ms
64 bytes from 10.13.16.170: icmp_seq=29 ttl=60 time=10.4 ms
no answer yet for icmp_seq=30
64 bytes from 10.13.16.170: icmp_seq=30 ttl=60 time=11.5 ms
64 bytes from 10.13.16.170: icmp_seq=31 ttl=60 time=10.9 ms
no answer yet for icmp_seq=32
64 bytes from 10.13.16.170: icmp_seq=32 ttl=60 time=11.5 ms
64 bytes from 10.13.16.170: icmp_seq=33 ttl=60 time=10.5 ms
no answer yet for icmp_seq=34
no answer yet for icmp_seq=35
64 bytes from 10.13.16.170: icmp_seq=34 ttl=60 time=24.7 ms
64 bytes from 10.13.16.170: icmp_seq=35 ttl=60 time=15.7 ms
64 bytes from 10.13.16.170: icmp_seq=36 ttl=60 time=13.5 ms
64 bytes from 10.13.16.170: icmp_seq=37 ttl=60 time=12.6 ms
64 bytes from 10.13.16.170: icmp_seq=38 ttl=60 time=10.5 ms
no answer yet for icmp_seq=39
64 bytes from 10.13.16.170: icmp_seq=39 ttl=60 time=13.7 ms
no answer yet for icmp_seq=40
64 bytes from 10.13.16.170: icmp_seq=40 ttl=60 time=14.2 ms
no answer yet for icmp_seq=41
64 bytes from 10.13.16.170: icmp_seq=41 ttl=60 time=14.4 ms
64 bytes from 10.13.16.170: icmp_seq=42 ttl=60 time=11.5 ms
64 bytes from 10.13.16.170: icmp_seq=43 ttl=60 time=11.7 ms
64 bytes from 10.13.16.170: icmp_seq=44 ttl=60 time=11.4 ms
```

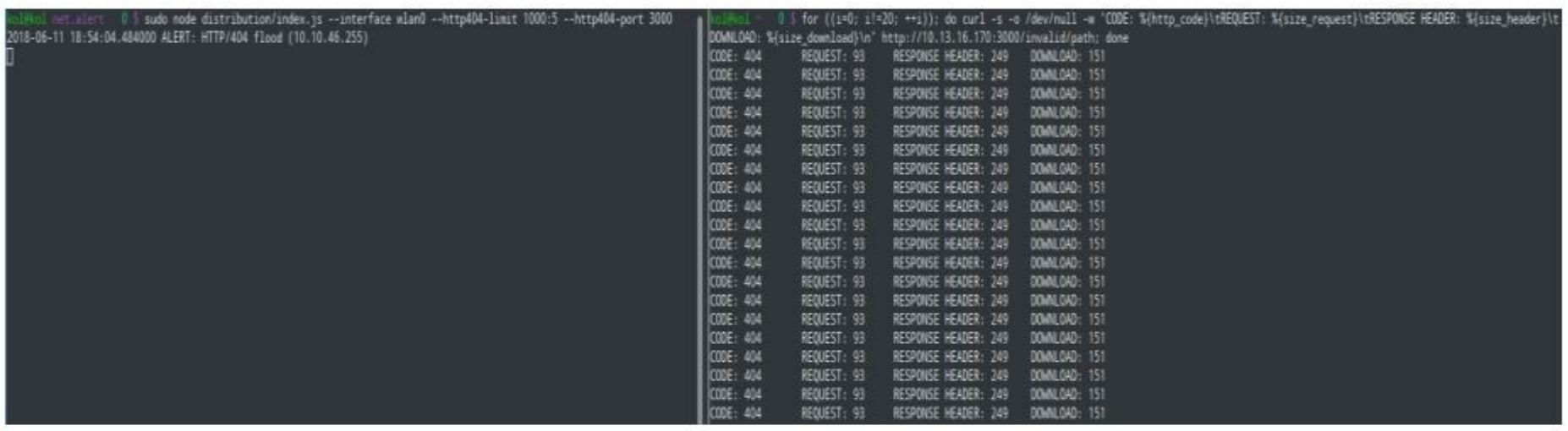
А Б

Рисунок 3.3 – Результат ICMP-flood атаки(А - виявлена атака та ip-адреса відправника пакетів), (Б – проведена атака)



А Б

Рисунок 3.4 – Результат SYN-flood атаки(А - виявлена атака та ip-адреса відправника пакетів), (Б – проведена атака)



А Б

Рисунок 3.5 - Результат HTTP-flood атаки(А - виявлена атака та ip-адреса відправника пакетів), (Б – проведена атака)

Далі для більш поглибленого тестування засобу було здійснено масивну DDoS-атаку на локальний сервер. Атака проводилась на однопроцесорний веб-сервер на базі 4-ядерної системи з багатьох одночасних потоків. Дана атака проводилась протягом 2417 секунд. Загалом було здійснено 3060075 запитів, тобто близько 1266 запитів в секунду. На графіках нижче зображено навантаження сервера в моент атаки.

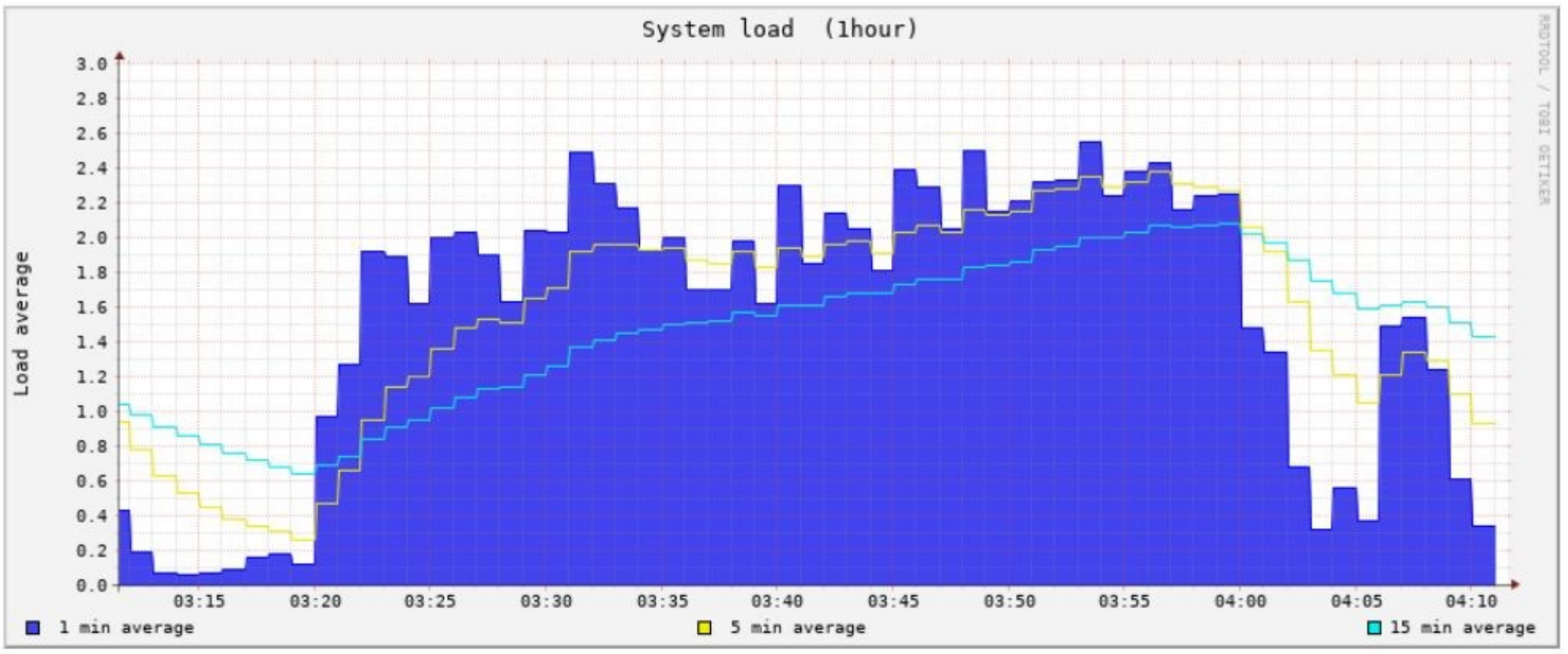


Рисунок 3.6 – Наватаження сервера протягом години

З даного графіка видно, що стабільне навантаження на сервер значно нижче пікових навантажень під час атаки. З цього можна зробити висновок, що сервер значно інтенсивніше використовує надані йому ресурси та погано справляється з наватаженням, що добре видно на наступному рисунку.

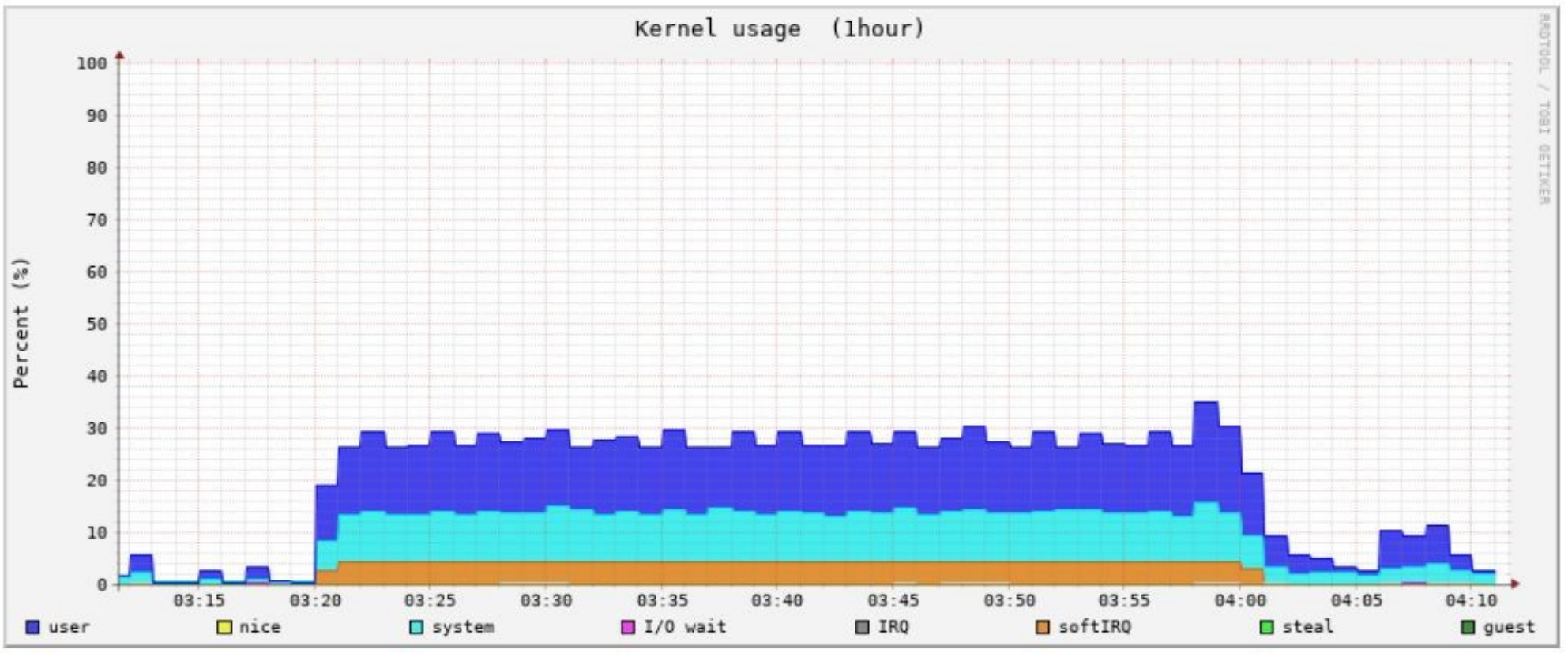


Рисунок 3.7 – Використання ресурсів процесора протягом години

На даному графіку видно як зросло користуваьке наватаження під час атаки, а отже можна зробити висновок, що сервер не справляється зі своєю задачею, так як намагається по декілька разів відповісти на вхідні запити. В середньому на один запит йшло біля 8 мілісекунд.

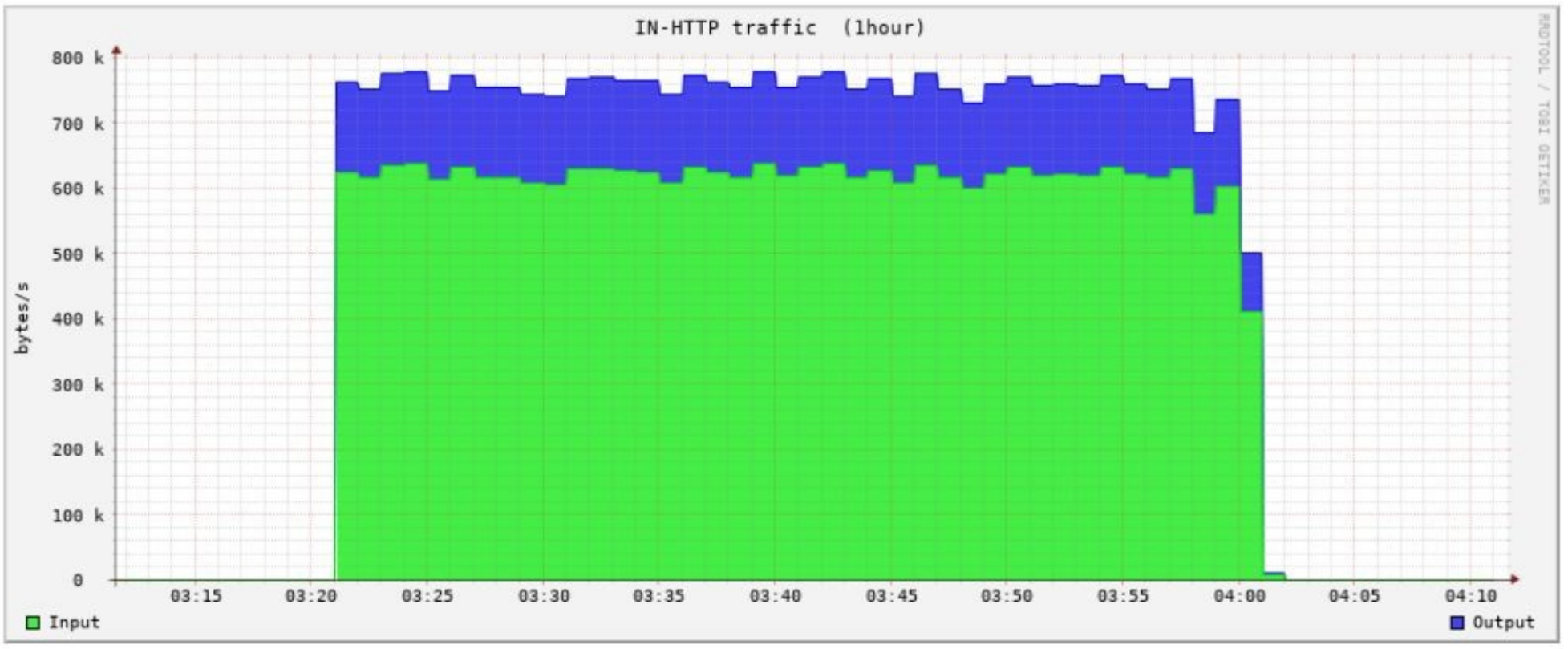


Рисунок 3.8 – Графік HTTP-трафіка протягом години

На даному графіку видно, що кількість вхідних запитів значно вище кількості вихідних, з чого можна зробити виснок, що сервер перевантажений і не встигає відповідати на всі запити вчасно. Дані дії приводять до того, що сервер відмовляється від виконання однієї задачі переходячи до іншої, що видно на рисунку нижче.

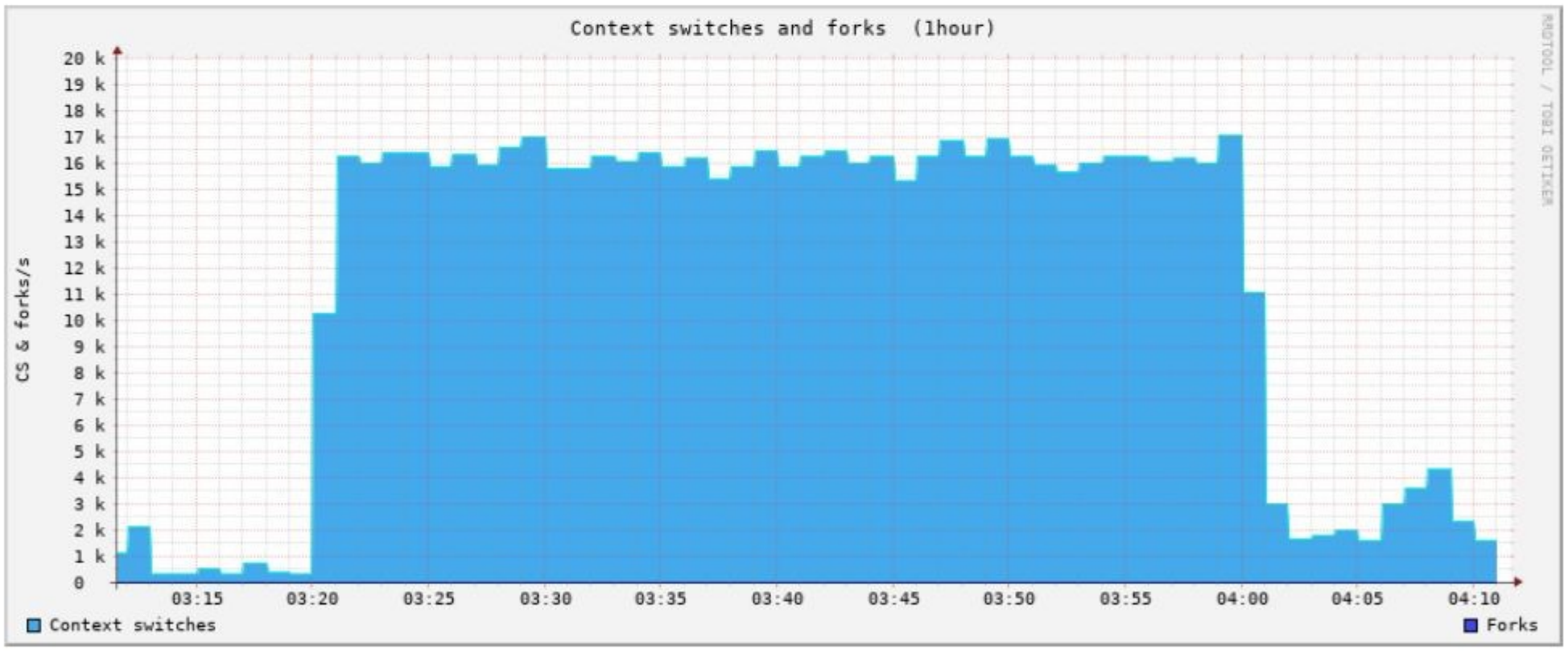


Рисунок 3.9 – Графік перемикання задач процесора

Даний графік показує кількість переходів сервера від однієї задачі до іншої, що у наслідку призводить до потребування значних ресурсів і загромадження його непотрібною інформацією, що в подальшому призведе до його відмови.

Розроблений програмний засіб виявив цю атаку ще на ранніх етапах і своєчасно сповістив користувача звідки йде загроза, тому при миттєвій реакції даних навантажень можна було б уникнути.

3.3 Порівння розробки з існуючими аналогами

Даний програмний засіб має багато аналогів, які вже давно розроблені, але потребують значно більше ресурсів від системи, в яку вони інтегровані. В порівнянні з іншими аналогами данний програмний засіб потребує значно менших ресурсів, має той же функціонал, але частина функцій прихована від рядового користувача, для зручності роботи та надання кращого киристувацького досвіду, шляхом виключення залишкової інформації. Розроблений програмний засіб добре справляється зі своєю функцією при екстремальних навантаженнях на сервер і своєчасно сповіщає користувача про можливу атаку. Це дозволяє уникнути перенавантежень на сервер безпосередньо на початок масової атаки.

Порівняльна характеристика розробленого методу виявлення з існуючими аналогами [19] відображена у таблиці 3.2.

Таблиця 3.2 – Порівння розробленого методу виявлення з аналогами

Метод виявлення атаки	Відсоток виявлення атаки	Хибна тривога	Можливість виявляти UDP/ICMP/HTTP атаки
CUSUM-SYN	100%	0%	Ні
CUSUM Ентропія	100%	14%	Так
4ЕКС Ентропія	100%	0%	Так

*всі параметри методу були оптимізовані для найкрощого виконання тестового сценарію

4ЕКС досягає відсотку виявлення CUSUM-SYN, але з іншого боку, вона має спільність CUSUM Ентропією.

Поєднання показників ентропії та швидкості пакетів дозволило запропонувати метод 4ЕКС, який є ефективним методом виявлення DDoS,

орієнтованим на більшість високошвидкісних та деяких низькошвидкісних DDoS-атак. Результати деяких високошвидкісних та низькошвидкісних атак наведені в таблиці 3.1. Це гібрид сигнатурного та статистичного методів виявлення. За допомогою динамічного прийняття рішень з поєднанням повільних і швидких тенденцій ЕКС, було встановлено пороги для автоматизованого механізму виявлення. За введення повільної тенденції в метод виявлення, стало можливим ігнорувати нерегулярні вибухи регулярного трафіку та встановлювати сигнал тривоги лише у випадку, якщо атака триватиме визначений час.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1. Аналіз комерційного потенціалу розробки (технологічний аудит розробки) гібридного методу виявлення DDoS-атак.

4.1 Визначення рівня комерційного потенціалу розробки гібридного методу виявлення DDoS-атак

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу гібридного методу виявлення DDoS-атак. В результаті оцінювання можна буде зробити висновок щодо напрямів (особливостей) організації подальшого її впровадження з врахуванням встановленого рейтингу.

Для проведення технологічного аудиту залучимо 3-х незалежних експертів. У нашому випадку такими експертами будуть керівник магістерської роботи та провідні викладачі випускової та споріднених кафедр.

Оцінювання комерційного потенціалу розробки гібридного методу виявлення DDoS-атак будемо здійснювати за 12-ю критеріями згідно рекомендацій.

Результати оцінювання комерційного потенціалу розробки гібридного методу виявлення DDoS-атак заносимо до таблиці 4.1.

Таблиця 4.1. - Результати оцінювання комерційного успіху гібридного методу виявлення DDoS-атак

Критерії	Експерти		
	д. т. н., проф., Лужецький В.А.	к.т.н., ст. викл Дудатьєв А. В.	к.т.н., доцент Войтович О.П.
	Бали, виставлені експертами		
1	2	2	3
2	3	3	2
3	4	4	3
4	4	2	3
5	4	3	3

Продовження Таблиці 4.1

6	4	4	3
7	3	3	2
8	3	3	3
9	3	4	4
10	2	3	3
11	3	3	3
12	2	3	3
Сума балів	37	37	35
Середньоарифметична сума балів, СБ	36		

За даними таблиці 4.1 робимо висновок щодо рівня комерційного потенціалу розробки гібридного методу виявлення DDoS-атак. При цьому користуємося рекомендаціями, наведеними в таблиці 4.2.

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 50	Високий

Таким чином, робимо висновок, щодо рівня комерційного потенціалу нашої розробки гібридного методу виявлення DDoS-атак вище середнього.

1.2 Визначення рівня якості розробки гібридного методу виявлення DDoS-атак

Оцінювання рівня якості розробки гібридного методу виявлення DDoS-атак проводиться з метою порівняльного аналізу і визначення найбільш ефективного, з технічної точки зору, варіанта інженерного рішення.

Рівень якості – це кількісна характеристика міри придатності певного виду продукції для задоволення конкретного попиту на неї при порівнянні з відповідними базовими показниками за фіксованих умов споживання.

Абсолютний рівень якості розробки адаптивного методу для автентифікації користувачів мобільних пристроїв знаходимо обчисленням вибраних для її вимірювання показників, не порівнюючи їх із відповідними показниками аналогічних виробів. Для цього необхідно визначити зміст основних функцій, які повинні реалізовувати розробка, вимоги замовника до неї, а також умови, які характеризують експлуатацію, визначають основні параметри, які будуть використані для розрахунку коефіцієнта технічного рівня виробу. Система параметрів, прийнята до розрахунків, повинна достатньо повно характеризувати споживчі властивості інноваційного товару (його призначення, надійність, економічне використання ресурсів, стандартизація тощо).

Далі визначаємо величину параметрів якості в балах та встановлюємо граничні його значення (кращі, гірші, середні). Всі ці дані для кожного параметра заносимо в табл. 4.3.

Таблиця 4.3 – Основні параметри гібридного методу виявлення DDoS-атак

Параметри	Абсолютне значення параметра			Коефіцієнт вагомості параметра
	Краще +5…+4	Середнє +3	Гірше +1…+2	

Низька відноснапохибка			2	0,2

Продовження Таблиці 4.3

Безпека			2	0,5
Ресурсозатратність		3		0,2
Варіативність		3		0,1

Із врахуванням коефіцієнтів вагомості відповідних параметрів можна визначити абсолютний рівень якості інноваційного рішення за формулою:

$$К_{я.а.} = \sum_{i=1}^{n} Рні \cdot аі, \quad (4.1)$$

де Рні – числове значення і-го параметра інноваційного рішення, n – кількість параметрів інноваційного рішення, що прийняті для оцінювання, аі – коефіцієнт вагомості відповідного параметра (сума коефіцієнтів вагомості всіх параметрів повинна дорівнювати 1).

Отже, абсолютний рівень якості методу та засобу завадостійкого розподілу секрету становитиме – 2,3 бали.

Одночасно визначаємо відносний рівень якості гібридного методу виявлення DDoS-атак, що виробляється (проектується), порівнюючи її показники з абсолютними показниками якості найліпших вітчизняних та зарубіжних аналогів (товарів-конкурентів) (табл. 4.4).

Таблиця 4.4 – Основні параметри гібридного методу виявлення DDoS-атак та товару-конкурента

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (конкурент)	Новий		

Продовження Таблиці 4.4

Низька відноснапохибка	2	1	2	0,2
Безпека	2	3	1,5	0,5
Ресурсозатратність	5	4	0,8	0,2
Варіативність	3	2	0,7	0,1

Відносний рівень якості методу та засобу завадостійкого розподілу секрету визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} qi \cdot ai, \quad (4.2)$$

За розрахунками відносний рівень якості гібридного методу виявлення DDoS-атак становитиме – 1,78. Це означає, що наша розробка краща за якістю на 78% від товару-аналога.

1.3 Визначення конкурентоспроможності розробки гібридного методу виявлення DDoS-атак

У найширшому розумінні конкурентоспроможність товару – це можливість його успішного продажу на певному ринку і в певний проміжок часу. Водночас конкурентоспроможною можна вважати лише однорідну продукцію з технічними параметрами і техніко-економічними показниками, що ідентичні аналогічним показникам уже проданого товару. Для того, щоб високоякісний товар був одночасно і конкурентоспроможним, він має відповідати критеріям оцінювання споживачів конкретного ринку в конкретний час.

Дані для розрахунку загального показника конкурентоспроможності розробки необхідно занести до таблиці 4.5.

Таблиця 4.5 – Нормативні, технічні та економічні параметри гібридного методу виявлення DDoS-атак і товару-конкурента

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (конкурент)	Новий		
Низька відноснапохибка	2	1	2	0,2
Безпека	2	3	1,5	0,5
Ресурсозатратність	5	4	0,8	0,2
Варіативність	3	2	0,7	0,1
Ціна за продукт, тис. грн.	3000	2000	0,7	-

Загальний показник конкурентоспроможності розробки (К) з урахуванням вищезазначених груп показників визначаємо за формулою:

$$К = \frac{Іт.п.}{Іе.п.} = \frac{1,78}{0,7} = 2,5, \quad (4.3)$$

де Іт.п. – індекс технічних параметрів (відносний рівень якості інноваційного рішення); Іе.п. – індекс економічних параметрів.

$$Іе.п. = \frac{РНеі}{РБеі} = \frac{2000}{3000} = 0,7, \quad (4.4)$$

де РНеі, РБеі – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

Згідно розрахунків загальний показник конкурентоспроможності –2,5. Це означає, що наша розробка гібридного методу виявлення DDoS-атак більш конкурентна майже в 2,5 рази від товару-аналога.

4.2. Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи

2.1 Розрахунок витрат, що стосуються виконавців розробки гібридного методу виявлення DDoS-атак

Основна заробітна плата кожного із розробників (дослідників) Зо, якщо вони працюють в наукових установах бюджетної сфери:

$$Зо = \frac{М}{Тр} \cdot t, \quad (4.5)$$

де М – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.

У 2019 році величини окладів (разом з встановленими доплатами і надбавками) рекомендується брати в межах (5000…10000) грн. за місяць; Тр – число робочих днів в місяці; приблизно Тр = (21…23) дні; t – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.6.

Таблиця 4.6 – Заробітна плата розробників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	8000	381	5	1905
Інженер-програміст	4500	214	5	1070
Всього:				2975

Основна заробітна плата робітників Зр, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$Зр = \sum_{i=1}^{n} ti \cdot Ci, \quad (4.6)$$

де ti – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; Сі – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. Сі визначається за формулою:

$$Сі = \frac{Мм \cdot Кі}{Тр \cdot Тзм}, \quad (4.7)$$

де Мм – розмір мінімальної заробітної плати за місяць, грн.; в 2019 році мінімальна заробітна плата становить – 4173 грн., Кі – тарифний коефіцієнт робітника відповідного розряду, Тр – число робочих днів в місяці; приблизно Тр = 21…23 дні; Тзм – тривалість зміни, зазвичай Тзм = 8 годин.

Таблиця 4.7 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Програмувальні	7	5	34	1,36	170
Всього					170

Додаткова заробітна плата Зд всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$Зд = 0{,}1 \cdot (Зр + Зо) = 0{,}1 \cdot (2975 + 170) = 314{,}5 \text{ грн.} \quad (4.8)$$

Нарахування на заробітну плату Нзп розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

де Зо – основна заробітна плата розробників, грн.; Зр – основна заробітна плата робітників, грн.; Зд – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

$$Нзп = 0{,}22 \cdot (Зр + Зо + Зд) = 0{,}22 \cdot (2975 + 170 + 314{,}5) = 761 \text{ грн.} \quad (4.9)$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц \cdot На}{100} \cdot \frac{Т}{12},$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; На – річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що На = (10...25)%; Т – термін, використання обладнання, приміщень тощо, місяці.

Таблиця 4.8 - Амортизаційні відрахування

Найменування	Ціна, грн.	Норма амортизації, %	Термін використання, м.	Сума амортизації
ПК	7000	20	2	233
Іншеобладнання	3000	10	1	25
Всього	258			

Витрати на силову електроенергію Ве, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$Ве = В \cdot П \cdot Ф \cdot Кп, грн$$

В – вартість 1 кВт-год. електроенергії, в 2019 р. В ≈ 8,45 грн./кВт; П – установлена потужність обладнання, кВт; Ф – фактична кількість годин роботи обладнання, годин, Кп – коефіцієнт використання потужності; Кп< 1.

Потужність обладнання складає –0,5 кВт.

Кількість годин роботи складає – 100годин.

Коефіцієнт викор. потужності -0,9.

$Ве= 8.45 \cdot 0,5 \cdot 100 \cdot 0,9 =$380,25 грн.

Інші витрати Він охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо.

Інші витрати Ів можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$Ів = 1{,}5 \cdot (Зо + Зр) = 1{,}5 \cdot (2975 + 170) = 4718 \text{ грн.} \quad (4.10)$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи – В.

$$В = 9577 \text{ грн.}$$

2.2 Розрахунок собівартості розробки гібридного методу виявлення DDoS-атак

Витрати на силову електроенергію Ве, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$Ве = В \cdot П \cdot Ф \cdot Кп, \text{ грн}$$

В – вартість 1 кВт-год. електроенергії, в 2019 р. В ≈ 8,45 грн./кВт; П – установлена потужність обладнання, кВт; Ф – фактична кількість годин роботи обладнання, годин, Кп – коефіцієнт використання потужності; Кп< 1.

Потужність обладнання складає – 0,5 кВт.

Кількість годин роботи складає – 100 годин.

Коефіцієнт викор. потужності - 0,9.

$$Ве = 8{,}45 \cdot 0{,}5 \cdot 100 \cdot 0{,}9 = 380{,}25 \text{ грн.}$$

Основна заробітна плата робітників Зр, якщо вони беруть участь у виконанні даного етапу роботи і виконують роботи за робочими професіями у випадку, коли вони працюють в наукових установах бюджетної сфери, розраховується за формулою:

$$Зр = \sum_{i=1}^{n} ti \cdot Ci, \quad (4.11)$$

де ti – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; Сі – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. Сі визначається за формулою:

$$Сі = \frac{Мм \cdot Кі}{Тр \cdot Тзм}, \quad (4.12)$$

де Мм – розмір мінімальної заробітної плати за місяць, грн.; в 2019 році мінімальна заробітна плата становить – 4173 грн., Кі – тарифний коефіцієнт робітника відповідного розряду, Тр – число робочих днів в місяці; приблизно Тр = 21…23 дні; Тзм – тривалість зміни, зазвичай Тзм = 8 годин.

Таблиця 4.9 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Програмувальні	7	5	34	1,36	170
Всього					170

Додаткова заробітна плата Зд всіх робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$Зд = 0{,}1 \cdot (Зо) = 0{,}1 \cdot (170) = 17 \text{ грн.} \quad (4.13)$$

Нарахування на заробітну плату Нзп розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

де Зо – основна заробітна плата розробників, грн.; Зр – основна заробітна плата робітників, грн.; Зд – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

$$Нзп = 0{,}22 \cdot (Зо + Зд) = 0{,}22 \cdot (170 + 17) = 41 \text{ грн.} \quad (4.14)$$

«Загальновиробничі витрати» належать витрати: пов'язані з управлінням виробництвом (утримання працівників апарату управління виробництвом, оплата службових відряджень персоналу цехів, витрати на інформаційне забезпечення

управління тощо); на повне відновлення та капітальний ремонт основних фондів загальновиробничого призначення; витрати некапітального характеру, пов'язані з удосконаленням технологій та організацією виробництва, поліпшенням якості продукції; на утримання, обслуговування, поточний ремонт виробничих приміщень; на контроль за виробничими процесами та кістю продукції.

Крім того, загальновиробничі витрати з розрахунку на одиницю продукції можна розрахувати за нормативами відносно до основної заробітної плати основних робітників, які виготовляють продукцію:

$$ЗВВ = Нв \cdot Зо, \quad (4.15)$$

Норматив загальновиробничих витрат для програмних продуктів становить 230-270%.

$$ЗВВ = 2{,}5 \cdot 170 = 425 \text{ грн},$$

Сума попередніх витрат утворює виробничу собівартість розробки:

$$Sв = 1033 \text{ грн}.$$

4.3. Розрахунок ціни та чистого прибутку від реалізації розробки гібридного методу виявлення DDoS-атак

Ціна – це грошовий вираз вартості товару (продукції, послуги). Вона завжди коливається навколо ціни виробництва (перетвореної форми вартості одиниці товару, що дорівнює сумі витрат виробництва й середнього прибутку) та відображає рівень суспільне необхідних витрат праці.

Виходячи з того, що розробки, як правило, приймаються та впроваджуються за завданням замовника, або коли результатом розробки є продукція, що підлягає державному регулюванню, то нижню межу ціни реалізації розробки можна розрахувати за формулою:

$$Ц = Sв \cdot \left(1 + \frac{Р}{100}\right) \cdot \left(1 + \frac{\omega}{100}\right), \quad (4.16)$$

де Sв – виробнича собівартість інноваційного рішення, грн.; Р – норматив рентабельності узгоджений із замовником або встановлений державою, (Р=30…60%); w – ставка податку на додану вартість, % (в 2019 році w=20%).

$$Ц = 1033 \cdot \left(1 + \frac{60}{100}\right) \cdot \left(1 + \frac{20}{100}\right) = 1983 \text{ грн.}$$

Чистий прибуток від реалізації розробки можна розрахувати за формулою:

$$П = \left(Ц - \frac{(Ц - МР) \cdot f}{100} - S_В - \frac{q \cdot S_В}{100}\right) \cdot \left(1 - \frac{h}{100}\right) \cdot РП, \quad (4.17)$$

де Ц – ціна розробки, грн.; МР – вартість матеріальних та інших ресурсів, що були придбані виробником для виготовлення розробки (МР=(0,1…0,2) Цр), грн.; f – зустрічна ставка податку на додану вартість, %; Sв – виробнича собівартість розробки, грн.; q – норматив, який визначає величину адміністративних витрат, витрат на збут та інші операційні витрати, % (рекомендовано q=5…10%); h – ставка податку на прибуток, %, РП – прогнозований попит продажів:

$$П = 8360 \text{ грн.}$$

4.4. Розрахунок терміну окупності коштів, вкладених в наукову розробку гібридного методу виявлення DDoS-атак

Термін окупності вкладених у реалізацію наукового проекту інвестицій Ток можна розрахувати за формулою:

$$Ток = \frac{В}{П} = \frac{9577}{8360} = 1{,}15 \text{ роки.} \quad (4.18)$$

Оскільки Ток < 3 років, то фінансування даної наукової розробки гібридного методу виявлення DDoS-атак.

ВИСНОВКИ

Магістрська робота присвячена проблемі виявлення DDoS-атак.

Аналіз інформаційних джерел показав, що існує велика кількість різного роду атак та небагато старих та непризначених програм для їх виявлення. Наведено класифікацію DDoS-атак, їх статистичні дані та методи їх реалізації. На основі цих даних було обрано атаки, які необхідно виявляти першочергово.

Другий розділ пояснювальної записки присвячений розробці методу виявлення атак та обґрунтуванню вибору технологій, які лягли в основу розробки, вибору мови програмування, та розробці програмного засобу для виявлення DDoS-атак.

Результатом виконання магістрської роботи є метод виявлення DDoS-атак та програмний засіб, що дозваляє практично використовувати даний метод, що виконаний у середовищі Visual Studio Code мовою JavaScript. Для розробки засобу підготовлений ряд схем і алгоритмів, розроблений інтерфейс, реалізовано цілу низку окремих функцій для виконання конкретних операцій. Розроблений програмний засіб протестований на предмет коректності роботи та ефективності виявлення різних типів атак.

Розроблений програмний засіб може бути використаний для захисту серверів від різних типів DDoS-атак, таких як ICMP-flood, SYN-flood та HTTP-flood.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кримінальний кодекс України : закон України від 23 грудня 2004 р. № 2289- IУ// Відомості Верховної Ради України. – 2004. – № 30. – Ст. 361.
2. Карта DDoS-атак в реальному часі [Електронний ресурс]. - Режим доступу : URL : http://www.digitalattackmap.com/_- Назва з екрану.
3. Крис Касперски, Андрей Комаров, Степан Ильин, Леонид Стройков, Сергей Яремчук, Денис Колесниченко. Хакер. — 2003.
4. W. Eddy. TCP SYN Flooding Attacks and Common Mitigations. — 2007.
5. Коллектив авторов. Иллюстрированный самоучитель по защите в Интернет. — 2004. — С. 2, 3, 4, 5, 6, 7, 8, 9, 12.
6. Мельников Д. А. Информационная безопасность открытых систем. — Москва: ФЛИНТА, 2012. — С. 448. — ISBN 978-5-9765-1613-7.
7. Виявлення DDoS-атак (DDoS) Офіційний сайт компанії Cisco [Електронний ресурс]. – Режим доступу : http://www.cisco.com/web/RU/products/ps5887/products_white_paper0900aecd8011e927_.html - Назва з екрану
8. Методы защиты от DDOS нападений [Електронний ресурс]. – Режим доступу: http://www.securitylab.ru/analytics/216251.php - Назва з екрану
9. Project Shield [Електронний ресурс]. – Режим доступу: https://projectshield.withgoogle.com/ru/ - Назва з екрану
10. Скляров И. Программирование боевого софта под Linux. —2007. — С. 35-59. — ISBN 5-94157-897-9.
11. ТАСС: Экономика и бизнес — «Лаборатория Касперского»: каждая шестая компания РФ в 2017 г. подвергалась DDoS-атаке.
12. Tcpdump [Електронний ресурс]. – Режим доступу: http://www.tcpdump.org/manpages/tcpdump.1.html - Назва з екрану.
13. Wireshark Q&A [Електронний ресурс]. - Режим доступу : URL : http://www.protocog.com/gerald_combs_interview.html - Назва з екрану.
14. Wireshark FAQ [Електронний ресурс]. - Режим доступу : URL : https://www.wireshark.org/faq.html#q1.2 - Назва з екрану.

15. Snort [Електронний ресурс]. – Режим доступу:https://www.snort.org/documents - Назва з екрану
16. Крис Касперски "Техника сетевых атак. Приёмы противодействия. Глава 'Что такое интернет' (Архитектура интернет. Дерево протоколов. Пакеты в Internet. Назначение портов.) "
17. Андрій Будай. Дизайн патерни – просто, як двері — Україна, 2012. — С. 52.
18. Pcap [Електронний ресурс]. – Режим доступу: http://www.tcpdump.org/#source - Назва з екрану
19. ddos attack scoreboard dataset, Mendeley Data, v2, [Електронний ресурс] - Режим доступу : URL :http://dx.doi.org/10.17632/ psjxnzsxyx.2 - Назва з екрану.

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

ЗАТВЕРДЖУЮ
д.т.н., проф. зав. кафедри ЗІ
_____ В. А.Лужецький
_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ
до магістерської кваліфікаційної роботи на тему
"Метод виявлення DDoS-атак"
08-20.МКР.007.00.000 ТЗ

Розробив студент групи 1БС-18м

_____ Ковальов В. А.

Керівник магістерської
кваліфікаційної роботи к. т. н., доц.
кафедри ЗІ

_____ Дудатьєв А. В.

_____ _____ 2019 р.

Вінниця 2019

1 Назва та область використання

Метод виявлення DDoS-атак. Область використання: захист від мережевих атак.

2 Підстави для розробки

Робота виконується згідно наказу №254 ректора ВНТУ від 02.10.2019 р.

3 Мета та призначення розробки

Зменшення похибки під час виявлення DDoS-атак.

4 Джерела розробки

4.1 W. Eddy. TCP SYN Flooding Attacks and Common Mitigations. — 2007.

4.2Коллектив авторов. Иллюстрированный самоучитель по защите в Интернет. — 2004. — С. 2, 3, 4, 5, 6, 7, 8, 9, 12.

4.3Мельников Д. А. Информационная безопасность открытых систем. — Москва: ФЛИНТА, 2012. — С. 448. — ISBN 978-5-9765-1613-7.

4.4ddos attack scoreboard dataset, Mendeley Data, v2, [Електронний ресурс] - Режим доступу : URL :http://dx.doi.org/10.17632/ psjxnzsxyx.2 - Назва з екрану.

5 Технічні вимоги

5.1 Метод виявлення – гібридний;
5.2 Середовище програмування – JavaScript;
5.3 Середовище тестування – реальна мережа.

6 Вимоги до програмної документації

5.1 Графічна і текстова документація повинна відповідати діючим стандартам України.
5.2 Метод повинен супроводжуватись:
- Алгоритмом методу виявлення DDoS-атак.
- Алгоритмом роботи програмного засобу.
- Структурою тестової мережі.
- Результатами тестування методу.

7 Стадії та етапи розробки

Робота по темі виконується у такі етапи:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.19 – 04.09.19	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	05.09.19 – 15.09.19	
3	Науково-технічне обґрунтування	16.09.19 – 21.09.19	
4	Розробка технічного завдання		
5	Розробка рішень	23.09.19 – 30.09.19	
6	Практична реалізація, моделювання, експериментування, результати	30.09.19 – 10.09.19	
7	Розробка розділу економічного обґрунтування доцільності розробки	14.10.19– 10.11.19	
8	Аналіз виконання ТЗ, висновки	11.11.19 – 17.11.19	
9	Оформлення пояснювальної записки	18.11.19 – 24.11.19	

10	Попередній захист та доопрацювання МКР	25.11.19 – 30.11.19	
11	Перевірка магістерської роботи на наявність плагіату	28.11.19 – 01.12.19	
12	Виправлення зауважень, підготовка ілюстративного матеріалу	02.12.19 – 10.12.19	
13	Представлення МКР до захисту, рецензування	11.12.19 – 14.12.19	
14	Захист МКР	16.12.19 – 20.12.19	

8 Порядок контролю та прийому

8.1 До приймання кваліфікаційної роботи представляється:

- ПЗ до магістерської кваліфікаційної роботи;
- робоча реалізація алгоритму;
- результати тестування;
- ілюстративні матеріали для захисту.

8.2 Рубіжний контроль керівника _____

8.3 Попередній захист на кафедрі 28 – 29 листопада

8.4 Захист на ДЕК 17 – 18 грудня

Розробив студент групи 1БС-18м _____ Ковальов В.А.

Додаток Б

Інструкція з технічного обслуговування

А.1 Вступ. Інструкція призначена для роз’яснення роботи програмного засобу для виявлення DDoS-атак.

А.2 Загальні вказівки. Програма призначена для виявлення DDoS атак.

А.3 Вимоги до технічних засобів. Мінімальний склад технічних засобів, потрібних для роботи програми:

- процесор – Pentium, Athlon і сумісні з ними;
- оперативна пам’ять – не менше 512 Мбайт;
- ОС Windows 7/ 8/10/Linux;
- монітор з розширенням 1024 x 768 та палітрою в 256 кольорів;
- маніпулятор “миша”;
- клавіатура.

А.4 Опис функцій. Програма виконує за допомогою оператора виконує визначених атак за певний період часу або без обмеження в часі.

Додаток В

Інструкція оператора

В.1 Призначення програми. Програма призначена для виявлення DDoS-атак.

В.2 Умови виконання програми. Апаратне забезпечення повинно відповідати такому як вказано в інструкції з технічного обслуговування (додаток А), встановлення і підготовка програми повинна відповідати інструкції системного адміністратора (додаток Б).

В.3 Виконання програми. Для виконання програми необхідно запустити виконуваний файл “net.alert та виконати роботу за допомогою команд.

В.4 Повідомлення оператору. Оператору будуть видаватися повідомлення про виявлену атаку та її ір-адресу у разі, якщо така здійснюється.

Додаток Г

Лістинг програми

Файл http.404.filter

```
import {LINK_TYPE} from "pcap2/decode";
import {IAlertEmitter} from '../alert/alert.emitter.interfacce';
import {TimeLimitedCounter} from '../time.limiter.counter';
import {APacketFilter} from './packet.filter.abstract';
import EthernetPacket = require('pcap2/decode/ethernet_packet');
import NullPacket = require('pcap2/decode/null_packet');
import RadioPacket = require('pcap2/decode/ieee802.11/radio_packet');
import SLLPacket = require('pcap2/decode/sll_packet');
import IPv4 = require('pcap2/decode/ipv4');
import IPv6 = require('pcap2/decode/ipv6');
import TCP = require('pcap2/decode/tcp');

export class Http404Filter extends APacketFilter {
    private readonly _counter: TimeLimitedCounter<string>;
    private readonly _ports: ReadonlySet<number>;

    public constructor(ports: ReadonlySet<number>, limits: Map<number,
number>, emitter?: IAlertEmitter) {
        super(emitter);
        this._counter = new TimeLimitedCounter<string>(limits);
        this._ports = new Set(ports.values());
    }

    public initialize(this: Http404Filter): void {

    }

    public finalize(this: Http404Filter): void {

    }

    public handle(this: Http404Filter, linkType: LINK_TYPE, packet:
EthernetPacket | NullPacket | RadioPacket | SLLPacket): boolean {
        if (!(packet instanceof RadioPacket) && packet.payload) {
            const networkLayer = packet.payload;
            if ((networkLayer instanceof IPv4 || networkLayer instanceof IPv6)
&& networkLayer.saddr && networkLayer.payload instanceof TCP) {
                const transportLayer = networkLayer.payload;
                if (this._ports.has(transportLayer.sport || 0) &&
transportLayer.dataLength && transportLayer.data) {
                    const link = `${networkLayer.daddr}`;
                    if
(/^HTTP[/]\d[.]\d\s+404/.test(transportLayer.data.toString()) &&
this._counter.add(link, 1)) {
                        this.alert({
                            message: 'HTTP/404 flood',
                            source: link,
                        });
                    }
                }
            }
        }
        return false;
    }
}
```

Файл icmp.flood.filter

```
import {LINK_TYPE} from "pcap2/decode";
```

```
import {IAlertEmitter} from '../alert/alert.emitter.interfacce';
import {TimeLimitedCounter} from '../time.limiter.counter';
import {APacketFilter} from './packet.filter.abstract';
import EthernetPacket = require('pcap2/decode/ethernet_packet');
import NullPacket = require('pcap2/decode/null_packet');
import RadioPacket = require('pcap2/decode/ieee802.11/radio_packet');
import SLLPacket = require('pcap2/decode/sll_packet');
import IPv4 = require('pcap2/decode/ipv4');
import IPv6 = require('pcap2/decode/ipv6');
import ICMP = require('pcap2/decode/icmp');

export class ICMPFloodFilter extends APacketFilter {
    private readonly _counter: TimeLimitedCounter<string>;

    public constructor(limits: Map<number, number>, emitter?: IAlertEmitter) {
        super(emitter);
        this._counter = new TimeLimitedCounter<string>(limits);
    }

    public initialize(this: ICMPFloodFilter): void {

    }

    public finalize(this: ICMPFloodFilter): void {

    }

    public handle(this: ICMPFloodFilter, linkType: LINK_TYPE, packet:
EthernetPacket | NullPacket | RadioPacket | SLLPacket): boolean {
        if (!(packet instanceof RadioPacket) && packet.payload) {
            const networkLayer = packet.payload;
            if ((networkLayer instanceof IPv4 || networkLayer instanceof IPv6)
&& networkLayer.saddr && networkLayer.payload instanceof ICMP) {
                const transportLayer = networkLayer.payload;
                if (transportLayer.type === 8 && transportLayer.code === 0) {
                    const link = `${networkLayer.saddr} =>
${networkLayer.daddr}`;
                    if (this._counter.add(link, 1)) {
                        this.alert({
                            message: 'ICMP flood',
                            source: link,
                        });
                    }
                    return true;
                }
            }
        }
        return false;
    }
}
```

Файл syn.flood.filter

```
import {LINK_TYPE} from "pcap2/decode";
import {IAlertEmitter} from '../alert/alert.emitter.interfacce';
import {TimeLimitedCounter} from '../time.limiter.counter';
import {APacketFilter} from './packet.filter.abstract';
import EthernetPacket = require('pcap2/decode/ethernet_packet');
import NullPacket = require('pcap2/decode/null_packet');
import RadioPacket = require('pcap2/decode/ieee802.11/radio_packet');
import SLLPacket = require('pcap2/decode/sll_packet');
import IPv4 = require('pcap2/decode/ipv4');
import IPv6 = require('pcap2/decode/ipv6');
```

```
import TCP = require('pcap2/decode/tcp');

export class SynFloodFilter extends APacketFilter {
    private readonly _counter: TimeLimitedCounter<string>;

    public constructor(limits: Map<number, number>, emitter?: IAlertEmitter) {
        super(emitter);
        this._counter = new TimeLimitedCounter<string>(limits);
    }

    public initialize(this: SynFloodFilter): void {

    }

    public finalize(this: SynFloodFilter): void {

    }

    public handle(this: SynFloodFilter, linkType: LINK_TYPE, packet:
EthernetPacket | NullPacket | RadioPacket | SLLPacket): boolean {
        if (!(packet instanceof RadioPacket) && packet.payload) {
            const networkLayer = packet.payload;
            if ((networkLayer instanceof IPv4 || networkLayer instanceof IPv6)
&& networkLayer.saddr && networkLayer.payload instanceof TCP) {
                const transportLayer = networkLayer.payload;
                if (transportLayer.flags && transportLayer.flags.syn) {
                    const link = `${networkLayer.saddr}`;
                    if (this._counter.add(link, 1)) {
                        this.alert({
                            message: 'SYN flood',
                            source: link,
                        });
                    }
                    return true;
                }
            }
        }
        return false;
    }
}
```

Файл pcap2.decode.d

```
declare module 'pcap2/decode/arp' {
    import {EventEmitter} from "events";
    import EthernetAddr = require('pcap2/decode/ethernet_addr');
    import IPv4Addr = require('pcap2/decode/ipv4_addr');

    class Arp {
        public constructor(emitter?: EventEmitter);
        public emitter?: EventEmitter;
        public htype?: number;
        public ptype?: number;
        public hlen?: number;
        public plen?: number;
        public operation?: number;
        public sender_ha?: EthernetAddr;
        public sender_pa?: IPv4Addr;
        public target_ha?: EthernetAddr;
        public target_pa?: IPv4Addr;
    }

    export = Arp;
}
```

```
declare module 'pcap2/decode/dns' {
    import {EventEmitter} from "events";
    import IPv4Addr = require('pcap2/decode/ipv4_addr');
    import IPv6Addr = require('pcap2/decode/ipv6_addr');

    class DnsFlags {
        public constructor();

        public isResponse?: boolean;
        public opcode?: number; // @todo const enum
        public isAuthority?: boolean;
        public isTruncated?: boolean;
        public isRecursionDesired?: boolean;
        public isRecursionAvailible?: boolean;
        public responseCode?: number; // @todo const enum

        public decode(this: DnsFlags, raw_packet: Buffer, offset: number):
this;
    }

    class DNSRR {
        public constructor(is_question: boolean);

        public name: string;
        public type?: number;
        public 'class'?: number;
        public ttl?: number;
        public rdlength?: number;
        public rdata?: IPv4Addr | IPv6Addr | string;
        public is_question: boolean;
    }

    class DNSRRSet {
        public constructor(count: number);

        public rrs: DNSRR[];
    }

    class DNS {
        public constructor(emitter?: EventEmitter);

        public emitter?: EventEmitter;
        public header?: DnsFlags;
        public qdcount?: number;
        public ancount?: number;
        public nscount?: number;
        public arcount?: number;
        public offset?: number;
        public question?: DNSRRSet;
        public answer?: DNSRRSet;
        public authority?: DNSRRSet;
        public additional?: DNSRRSet;

        public decode(this: DNS, raw_packet: Buffer, offset: number): this;
        public decode_RRs(this: DNS, count: number, is_question: boolean):
DNSRRSet;
        public decode_RR(this: DNS, is_question: boolean): DNSRR;
        public read_name(this: DNS): string;
    }

    export = DNS;
}
```

```
declare module 'pcap2/decode/ethernet_addr' {
    class EthernetAddr {
        public constructor(raw_packet: Buffer, offset: number);
        addr: [number, number, number, number, number, number];
    }

    export = EthernetAddr;
}

declare module 'pcap2/decode/ethernet_packet' {
    import {EventEmitter} from 'events';
    import EthernetAddr = require('pcap2/decode/ethernet_addr');
    import Vlan = require('pcap2/decode/vlan');
    import IPv4 = require('pcap2/decode/ipv4');
    import IPv6 = require('pcap2/decode/ipv6');
    import Arp = require('pcap2/decode/arp');

    class EthernetPacket {
        public constructor(emitter?: EventEmitter);

        public emitter?: EventEmitter;
        public dhost?: EthernetAddr;
        public shost?: EthernetAddr;
        public ethertype?: number; /// @todo const enum ETHER_TYPE
        public vlan: Vlan;
        public payload: IPv4 | Arp | IPv6;

        public decode(this: EthernetPacket, raw_packet: Buffer, offset:
number);
    }

    export = EthernetPacket;
}

declare module 'pcap2/decode/icmp' {
    import {EventEmitter} from 'events';

    class ICMP {
        public constructor(emitter?: EventEmitter);

        public emitter?: EventEmitter;
        public type?: number; // @todo const enum
        public code?: number; // @todo const enum
        public checksum?: number;

        public decode(this: ICMP, raw_packet: Buffer, offset: number);
    }

    export = ICMP;
}

declare module 'pcap2/decode/igmp' {
    import {EventEmitter} from 'events';
    import IPv4Addr = require('pcap2/decode/ipv4_addr');

    class IGMP {
        public constructor(emitter?: EventEmitter);

        public emitter?: EventEmitter;
        public type?: number; // @todo const enum
        public version?: number;
        public maxResponseTime?: number;
```

```
        public checksum?: number;
        public groupAddress?: IPv4Addr;

        public decode(this: IGMP, raw_packet: Buffer, offset: number);
    }

    export = IGMP;
}

declare module 'pcap2/decode/ip_protocols' {
    import ICMP = require('pcap2/decode/icmp');
    import IGMP = require('pcap2/decode/igmp');
    import IPv4 = require('pcap2/decode/ipv4');
    import IPv6 = require('pcap2/decode/ipv6');
    import                              IPV6HeaderExtension                              =
require('pcap2/decode/ipv6headers/header_extension');
    import TCP = require('pcap2/decode/tcp');
    import UDP = require('pcap2/decode/udp');
    import NoNext = require('pcap2/decode/ipv6headers/no_next');

    interface IClass<InstanceType> {
        new (): InstanceType;
    }

    interface IProtocols {
        0: IClass<IPV6HeaderExtension>;
        1: IClass<ICMP>;
        2: IClass<IGMP>;
        4: IClass<IPv4>;
        6: IClass<TCP>;
        17: IClass<UDP>;
        41: IClass<IPv6>;
        43: IClass<IPV6HeaderExtension>;
        51: IClass<IPV6HeaderExtension>;
        59: IClass<NoNext>;
        60: IClass<IPV6HeaderExtension>;
        135: IClass<IPV6HeaderExtension>;
        139: IClass<IPV6HeaderExtension>;
        140: IClass<IPV6HeaderExtension>;

        [key: number]: IClass<IPV6HeaderExtension | ICMP | IGMP | IPv4 | TCP |
UDP | IPv6 | NoNext>;
    }

    export = IProtocols;
}

declare module 'pcap2/decode/ipv4' {
    import {EventEmitter} from 'events';
    import IPv4Addr = require('pcap2/decode/ipv4_addr');
    import ICMP = require('pcap2/decode/icmp');
    import IGMP = require('pcap2/decode/igmp');
    import TCP = require('pcap2/decode/tcp');
    import UDP = require('pcap2/decode/udp');

    class IPFlags {
        reserved?: boolean;
        doNotFragment?: boolean;
        moreFragments?: boolean;

        decode(this: IPFlags, raw_flags: number): this;
    }
```

```
class IPv4 {
    public constructor(emitter?: EventEmitter);

    public emitter?: EventEmitter;
    public version?: number;
    public headerLength?: number;
    public diffserv?: number;
    public length?: number;
    public identification?: number;
    public flags?: IPFlags;
    public fragmentOffset?: number;
    public ttl?: number;
    public protocol?: number; /// @todo const enum
    public headerChecksum?: number;
    public saddr?: IPv4Addr;
    public daddr?: IPv4Addr;
    public payload?: ICMP | IGMP | TCP | UDP;

    public decode(this: IPv4, raw_packet: Buffer, offset: number): this;
  }

  export = IPv4;
}

declare module 'pcap2/decode/ipv4_addr' {
  class IPv4Addr {
    public constructor();
    public addr: [number, number, number, number];

    public  decode(this:  IPv4Addr,  raw_packet:  Buffer,  offset:  number):
this;
  }

  export = IPv4Addr;
}

declare module 'pcap2/decode/ipv6' {
  import {EventEmitter} from 'events';
  import IPv6Addr = require('pcap2/decode/ipv6_addr');
  import ICMP = require('pcap2/decode/icmp');
  import IGMP = require('pcap2/decode/igmp');
  import TCP = require('pcap2/decode/tcp');
  import UDP = require('pcap2/decode/udp');

  class IPv6 {
    public constructor(emitter?: EventEmitter);

    public emitter?: EventEmitter;
    public version?: number;
    public trafficClass?: number; /// @todo const enum
    public flowLabel?: number; /// @todo const enum
    public payloadLength?: number;
    public nextHeader?: number;
    public hopLimit?: number;
    public saddr?: IPv6Addr;
    public daddr?: IPv6Addr;
    public payload?: ICMP | IGMP | TCP | UDP;

    public decode(this: IPv6, raw_packet: Buffer, offset: number): this;
  }

  export = IPv6;
}
```

```
        declare module 'pcap2/decode/ipv6_addr' {
            class IPv6Addr {
                public constructor();
                public addr: [  number, number, number, number,
                                number, number, number, number,
                                number, number, number, number,
                                number, number, number, number  ];

                public  decode(this:  IPv6Addr,  raw_packet:  Buffer,  offset:  number):
this;
            }

            export = IPv6Addr;
        }

        declare module 'pcap2/decode/llc_packet' {
            import {EventEmitter} from 'events';
            import IPv4 = require('pcap2/decode/ipv4');

            class LogicalLinkControl {
                public constructor(emitter?: EventEmitter);

                public emitter?: EventEmitter;
                public dsap?: number; /// @todo const enum
                public ssap?: number; /// @todo const enum
                public controlField?: number;
                public orgCode?: [number, number, number];
                public type?: number; /// @todo const enum
                public payload?: IPv4;

                public  decode(this:  LogicalLinkControl,  raw_packet:  Buffer,  offset:
number): this;
            }

            export = LogicalLinkControl;
        }

        declare module 'pcap2/decode/null_packet' {
            import {EventEmitter} from 'events';
            import IPv4 = require('pcap2/decode/ipv4');
            import IPv6 = require('pcap2/decode/ipv6');

            class NullPacket {
                public constructor(emitter?: EventEmitter);

                public emitter?: EventEmitter;
                public pftype?: number; /// @todo const enum
                public payload?: IPv4 | IPv6;

                public  decode(this:  NullPacket,  raw_packet:  Buffer,  offset:  number):
this;
            }

            export = NullPacket;
        }

        declare module 'pcap2/decode/sll_addr' {
            class SLLAddr {
                public constructor(raw_packet: Buffer, offset: number, len: number);
                public addr: number[];
            }
```

```
    export = SLLAddr;
}

declare module 'pcap2/decode/sll_packet' {
    import {EventEmitter} from 'events';
    import IPv4 = require('pcap2/decode/ipv4');
    import IPv6 = require('pcap2/decode/ipv6');
    import SLLAddr = require('pcap2/decode/sll_addr');
    import Arp = require('pcap2/decode/arp');

    class SLLPacket {
        public constructor(emitter?: EventEmitter);

        public emitter?: EventEmitter;
        public packet_type?: number; /// @todo const enum
        public address_type?: number; /// @todo const enum
        public address_len?: number; /// @todo const enum
        public address?: SLLAddr;
        public ethertype?: number; /// @todo const enum

        public payload?: IPv4 | IPv6 | Arp | string;

        public decode(this: SLLPacket, raw_packet: Buffer, offset: number):
this;
    }

    export = SLLPacket;
}

declare module 'pcap2/decode/tcp' {
    import {EventEmitter} from 'events';

    class TCPFlags {
        public constructor(emitter: EventEmitter);

        public emitter?: EventEmitter;
        public nonce?: boolean;
        public cwr?: boolean;
        public ece?: boolean;
        public urg?: boolean;
        public ack?: boolean;
        public psh?: boolean;
        public rst?: boolean;
        public syn?: boolean;
        public fin?: boolean;

        public decode(this: TCPFlags, first_byte: number, second_byte:
number): this;
    }

    class TCPOptions {
        public constructor();

        public mss?: number;
        public window_scale: number;
        public sack_ok?: boolean;
        public sack?: number[];
        public timestamp?: number;
        public echo?: number;

        public decode(this: TCPOptions, raw_packet: Buffer, offset: number,
len: number);
    }
```

```
class TCP {
    public constructor(emitter: EventEmitter);

    public emitter?: EventEmitter;
    public sport?: number;
    public dport?: number;
    public seqno?: number;
    public ackno?: number;
    public headerLength?: number;
    public flags?: TCPFlags;
    public windowSize?: number;
    public checksum?: number;
    public urgentPointer?: number;
    public options?: TCPOptions;
    public dataLength?: number;
    public data?: Buffer;

    public decode(this: TCP, raw_packet: Buffer, offset: number, len:
number): this;
}

export = TCP;
}

declare module 'pcap2/decode/udp' {
    import {EventEmitter} from 'events';

    class UDP {
        public constructor(emitter: EventEmitter);
        public emitter?: EventEmitter;
        public sport?: number;
        public dport?: number;
        public length?: number;
        public checksum?: number;
        public data?: Buffer;

        public decode(this: UDP, raw_packet: Buffer, offset: number, len:
number): this;
    }

    export = UDP;
}

declare module 'pcap2/decode/vlan' {
    class Vlan {
        public constructor();
        public priority?: number;
        public canonical_format?: number;
        public id?: number;

        public decode(this: Vlan, raw_packet: Buffer, offset: number): this;
    }

    export = Vlan;
}

declare module 'pcap2/decode/ipv6headers/header_extension' {
    import TCP = require('pcap2/decode/tcp');
    import UDP = require('pcap2/decode/udp');
```

```
            class HeaderExtension {
                public constructor();

                public payload?: TCP | UDP;
                public nextHeader?: number;
                public headerLength?: number;

                public  decode(this:  HeaderExtension,  raw_packet:  Buffer,  offset:
number): this;
            }

            export = HeaderExtension;
        }

        declare module 'pcap2/decode/ipv6headers/no_next' {
            class NoNext {
                public constructor();

                protected _error?: string;

                public decode(this: NoNext, raw_packet: Buffer, offset: number): this;
            }

            export = NoNext;
        }

        declare module 'pcap2/decode/ieee802.11/radio_beacon_frame' {
            import                          RadioMangementFrameTag                          =
require('pcap2/decode/ieee802.11/radio_management_fragment_tag');

            class RadioBeaconFrame {
                public tags: RadioMangementFrameTag[];

                public  decode(this:  RadioBeaconFrame,  raw_packet:  Buffer,  offset:
number): this;
            }

            export = RadioBeaconFrame;
        }

        declare module 'pcap2/decode/ieee802.11/radio_frame' {
            import {EventEmitter} from 'events';
            import EthernetAddr = require('pcap2/decode/ethernet_addr');
            import                              RadioBeaconFrame                            =
require('pcap2/decode/ieee802.11/radio_beacon_frame');
            import LogicalLinkControl = require('pcap2/decode/llc_packet');
            import                              RadioProbeFrame                             =
require('pcap2/decode/ieee802.11/radio_probe_frame');

            class RadioFrameFlags {
                public constructor(emitter?: EventEmitter);

                public emitter?: EventEmitter;
                public raw?: number;
                public moreFragments?: boolean;
                public isRetry?: boolean;
                public moreData?: boolean;
                public encrypted?: boolean;
                public ordered?: boolean;

                public  decode(this:  RadioFrameFlags,  raw_packet:  Buffer,  offset:
number): this;
            }
```

```
class RadioFrame {
    public constructor(emitter?: EventEmitter);

    public emitter?: EventEmitter;
    public frameControl?: number;
    public version?: number;
    public type?: number;
    public subType?: number;
    public flags?: RadioFrameFlags;
    public duration?: number;
    public bssid?: EthernetAddr;
    public shost?: EthernetAddr;
    public dhost?: EthernetAddr;
    public fragSeq?: number;

    public probe?: RadioProbeFrame;
    public beacon?: RadioBeaconFrame;
    public llc?: LogicalLinkControl;

    public decode(this: RadioFrame, raw_packet: Buffer, offset: number):
this;
}

export = RadioFrame;
}

declare module 'pcap2/decode/ieee802.11/radio_management_fragment_tag' {
    class RadioMangementFrameTag {
        public constructor();
        public type?: string;
        public typeId?: number;
        public length?: number;
        public value?: Buffer;

        public decode(this: RadioMangementFrameTag, raw_packet: Buffer,
offset: number): this;
    }

    export = RadioMangementFrameTag;
}

declare module 'pcap2/decode/ieee802.11/radio_packet' {
    import {EventEmitter} from 'events';
    import RadioFrame = require('pcap2/decode/ieee802.11/radio_frame');

    class PresentFieldFlags {
        public constructor();

public tsft?: boolean;
public flags?: boolean;
public rate?: boolean;
public channel?: boolean;
public fhss?: boolean;
public signalStrength?: boolean;
public signalNoise?: boolean;
public lockQuality?: boolean;
public txAttenuation?: boolean;
public dbTxAttenuation?: boolean;
public dbmTxPower?: boolean;
public antenna?: boolean;
public dbAntennaSignal?: boolean;
```

```
            public dbAntennaNoise?: boolean;
            public rxFlags?: boolean;

            public  decode(this:  PresentFieldFlags,  raw_packet:  Buffer,  offset:
number): this;
        }

        class RadioPacket {
            public constructor(emitter?: EventEmitter);

            public emitter?: EventEmitter;
            public headerRevision?: number;
            public headerPad?: number;
            public headerLength?: number;
            public signalStrength?: number;
            public frequency?: number;
            public antenna?: number;
            public ieee802_11Frame?: RadioFrame;
            public presentFields?: PresentFieldFlags;
            public signalNoise?: number;

            public decode(this: RadioPacket, raw_packet: Buffer, offset: number):
this;
        }

        export = RadioPacket;
    }

    declare module 'pcap2/decode/ieee802.11/radio_probe_frame' {
        import                          RadioMangementFrameTag                          =
require('pcap2/decode/ieee802.11/radio_management_fragment_tag');

        class RadioProbeFrame {
            public tags: RadioMangementFrameTag[];

            public  decode(this:  RadioProbeFrame,  raw_packet:  Buffer,  offset:
number): this;
        }

        export = RadioProbeFrame;
    }

    declare module 'pcap2/decode/ieee802.11/radio_utils' {
        import                          RadioMangementFrameTag                          =
require('pcap2/decode/ieee802.11/radio_management_fragment_tag');

        export  function  parseTags(raw_packet:  Buffer,  offset:  number):
RadioMangementFrameTag[];
    }

declare module 'pcap2/decode/pcap_packet' {
    import {EventEmitter} from 'events';
    import {LINK_TYPE} from 'pcap2/decode';
    import EthernetPacket = require('pcap2/decode/ethernet_packet');
    import NullPacket = require('pcap2/decode/null_packet');
    import RadioPacket = require('pcap2/decode/ieee802.11/radio_packet');
    import SLLPacket = require('pcap2/decode/sll_packet');

    class PcapHeader {
        public constructor(raw_header: Buffer);

        readonly tv_sec: number;
        readonly tv_usec: number;
```

```
            readonly caplen: number;
            readonly len: number;
        }

        class PcapPacket {
            public constructor(emitter?: EventEmitter);

            public emitter?: EventEmitter;
            public link_type?: LINK_TYPE;
            public pcap_header?: PcapHeader;
            public  payload?:  EthernetPacket  |  NullPacket  |  RadioPacket  |
SLLPacket;

            public decode(this: PcapPacket, raw_packet: Buffer, offset: number):
this;
        }

        export = PcapPacket;
    }

    declare module 'pcap2/decode' {
        import {EventEmitter} from 'events';
        export import EthernetPacket = require('pcap2/decode/ethernet_packet');
        export import IPv4 = require('pcap2/decode/ipv4');
        export import IPv6 = require('pcap2/decode/ipv6');
        export import Arp = require('pcap2/decode/arp');
        export import PcapPacket = require('pcap2/decode/pcap_packet');

        const enum LINK_TYPE {
            ETHERNET = 'LINKTYPE_ETHERNET',
            NULL = 'LINKTYPE_NULL',
            RAW = 'LINKTYPE_RAW',
            IEEE802_11_RADIO = 'LINKTYPE_IEEE802_11_RADIO',
            LINUX_SLL = 'LINKTYPE_LINUX_SLL',
        }

        export interface IPacketWithHeader {
            buf: Buffer;
            header: Buffer;
            link_type: LINK_TYPE;
        }

        export function decode(packet: IPacketWithHeader, emitter?: EventEmitter):
PcapPacket;
    }
```

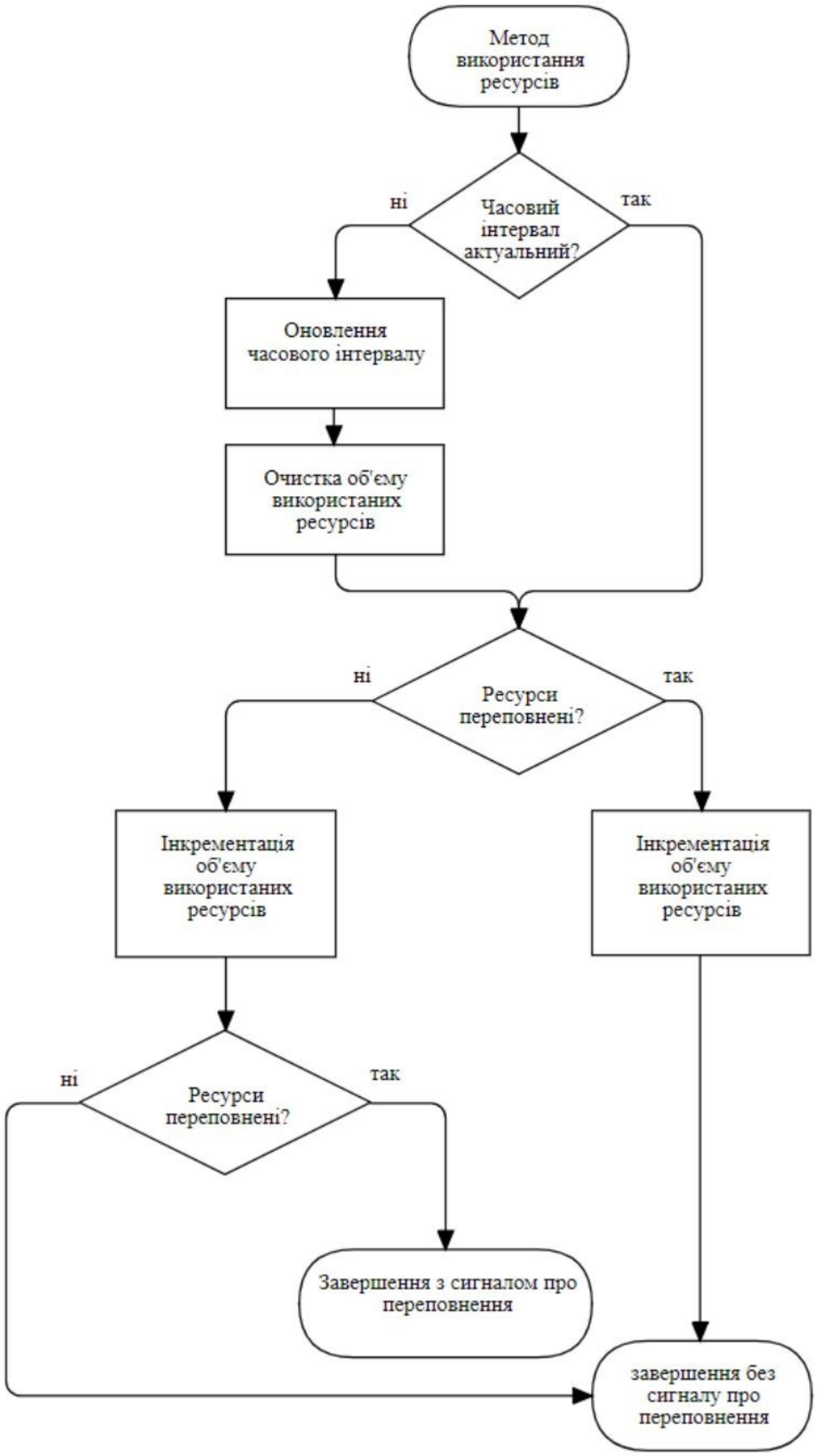
Файл pcap2.d

```
interface ISessionOptions {
    bufferSize: number;
    isMonitor: boolean;
    outfile: string;
    filter: string;
    timeout: number;
}

interface ISessionStats {
    ps_recv: number;
    ps_drop: number;
    ps_ifdrop: number;
}

declare module 'pcap2/lib/session' {
```

```
            import {EventEmitter} from 'events';
            import {IPacketWithHeader} from 'pcap2/decode';

            class Session extends EventEmitter {
                public readonly fd: number;
                public readonly opened: boolean;
                public readonly buf: Buffer;
                public readonly header: Buffer;
                public readonly options: Readonly<ISessionOptions>;

                public constructor(is_live: boolean, device_name: string, options?:
Partial<ISessionOptions>);

                public close(this: Session): void;
                public stats(this: Session): ISessionStats;
                public inject(this: Session, data: Buffer): void;

                public addListener(event: 'packet', handler: (packet:
IPacketWithHeader) => void): this;
                public addListener(event: string | symbol, handler: (...args: any[])
=> void): this;

                public on(event: 'packet', handler: (packet: IPacketWithHeader) =>
void): this;
                public on(event: string | symbol, handler: (...args: any[]) => void):
this;

                public once(event: 'packet', handler: (packet: IPacketWithHeader) =>
void): this;
                public once(event: string | symbol, handler: (...args: any[]) =>
void): this;

                public prependListener(event: 'packet', handler: (packet:
IPacketWithHeader) => void): this;
                public prependListener(event: string | symbol, handler: (...args:
any[]) => void): this;

                public prependOnceListener(event: 'packet', handler: (packet:
IPacketWithHeader) => void): this;
                public prependOnceListener(event: string | symbol, handler: (...args:
any[]) => void): this;

                public removeListener(event: 'packet', handler: (packet:
IPacketWithHeader) => void): this;
                public removeListener(event: string | symbol, handler: (...args:
any[]) => void): this;
            }

            export = Session;
        }

        declare module 'pcap2/lib/liveSession' {
            import Session = require('pcap2/lib/session');

            class LiveSession extends Session {
                public constructor(device_name: string, options?:
Partial<ISessionOptions>);
            }

            export = LiveSession;
        }

        declare module 'pcap2/lib/offlineSession' {
```

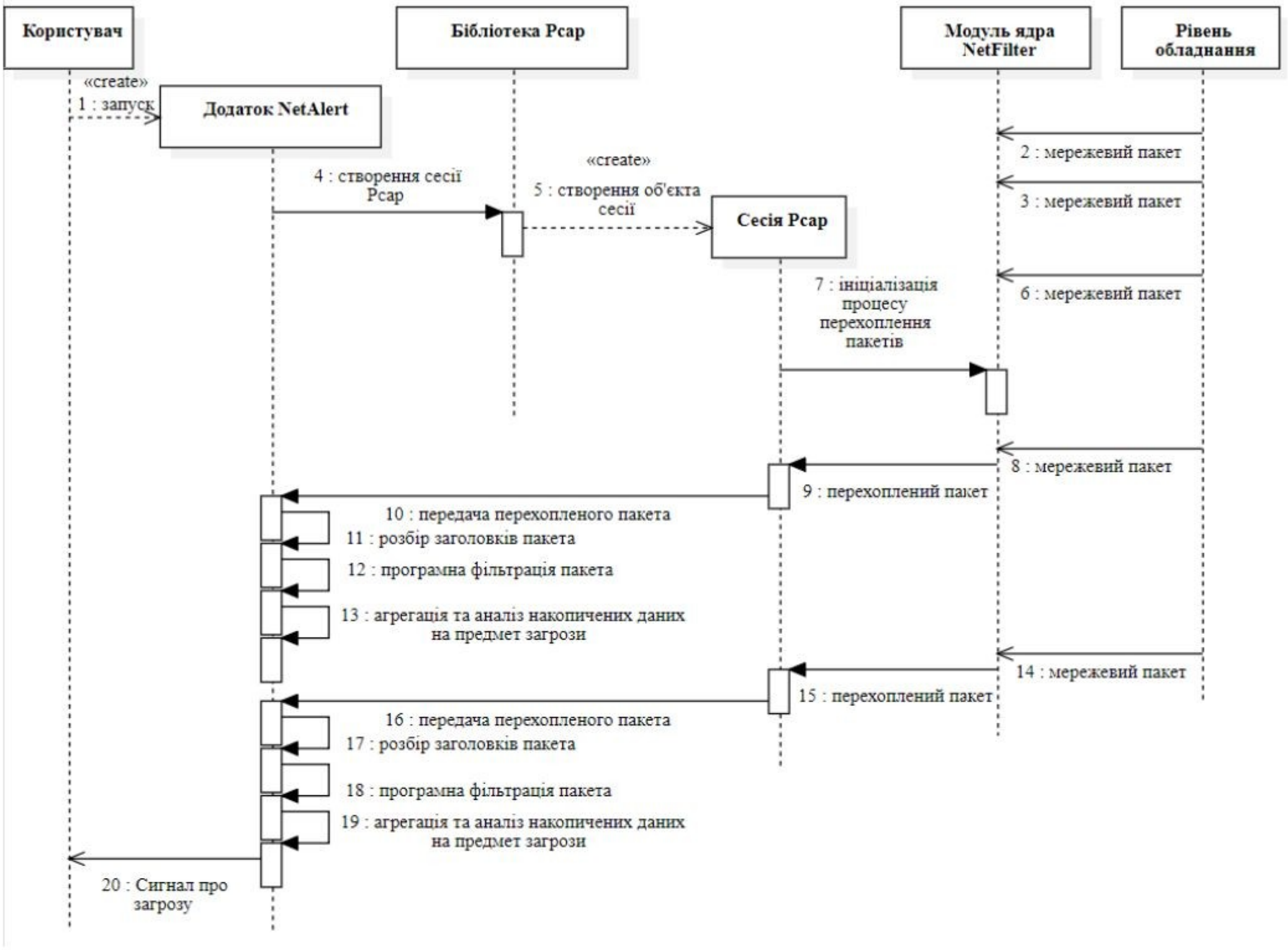
```
    import Session = require('pcap2/lib/session');

    class OfflineSession extends Session {
        public constructor(path: string, options?: Partial<ISessionOptions>);
    }

    export = OfflineSession;
}

/// @todo declare module 'pcap2/lib/tcpTracker'
/// @todo declare module 'pcap2/lib/dnsCache'

declare module 'pcap2' {
    export import Session = require('pcap2/lib/liveSession');
    export import OfflineSession = require('pcap2/lib/offlineSession');
    export {decode} from 'pcap2/decode';
    /// @todo TCPSession
    /// @todo TCPTracker
    /// @todo DNSCache
}
```

ЛІЧИЛЬНИК З ОБМЕЖЕННЯМ ПО РЕСУРСАМ



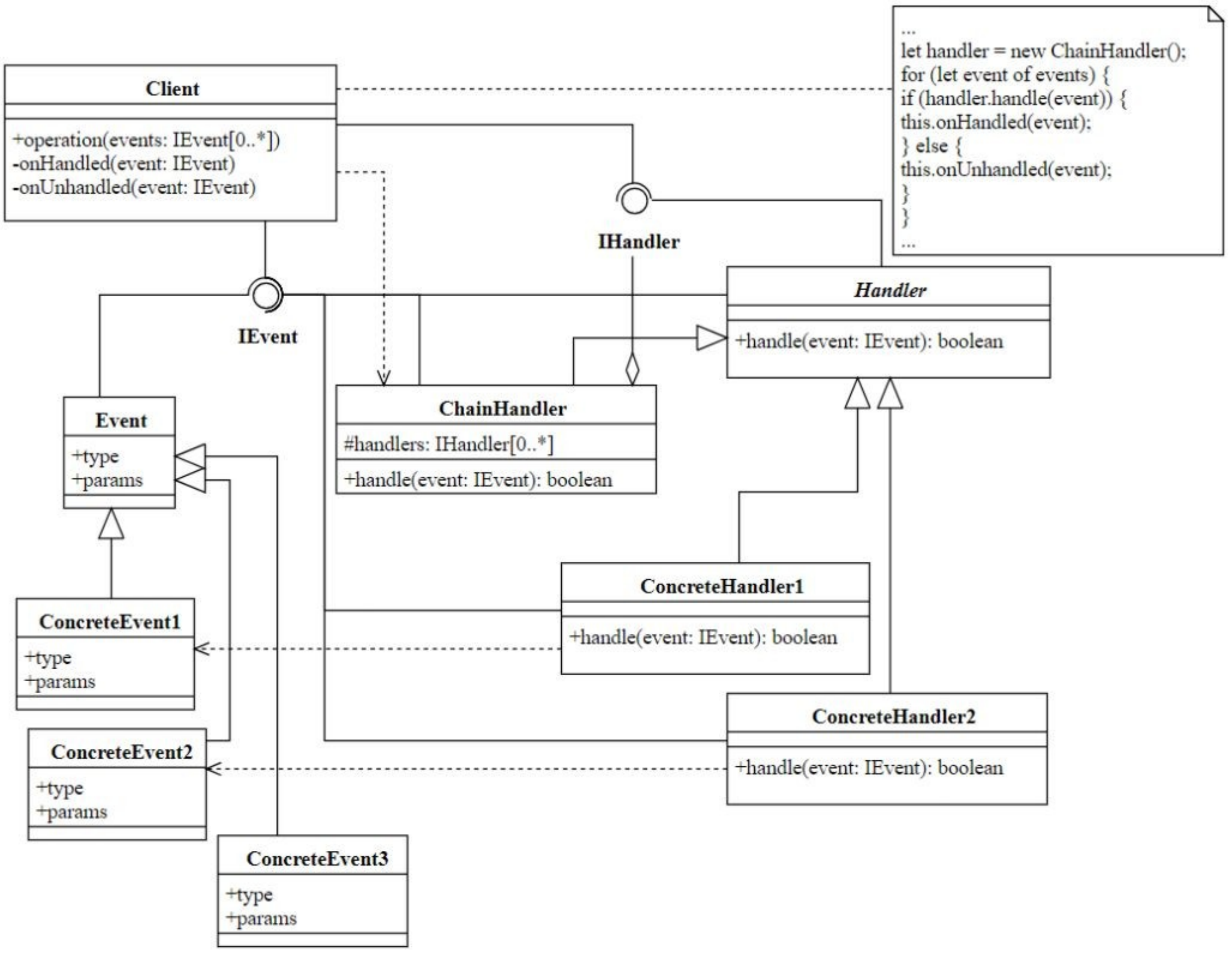
					08-20.МКР.007.00.000 ІЧ5			
Змн	Арк.	№ докум.	Підпис	Дат				
Розроб.		Ковальов В.А.			Метод виявлення DDoS-атак. Лічильник з обмеженням по ресурсам	Літ.	Арк.	Аркушів
Перевір.		Дудатьєв А.В.					1	1
Реценз.		КрупельницькийЛ.				ВНТУ, гр.1БС-18м		
Н. Контр.		Дудатьєв А.В.						
Затверд.		Лужецький В.А.						

СХЕМА ПОСЛІДОВНОСТІ ВЗАЄМОДІЇ ПРОГРАМНОГО ЗАСОБУ



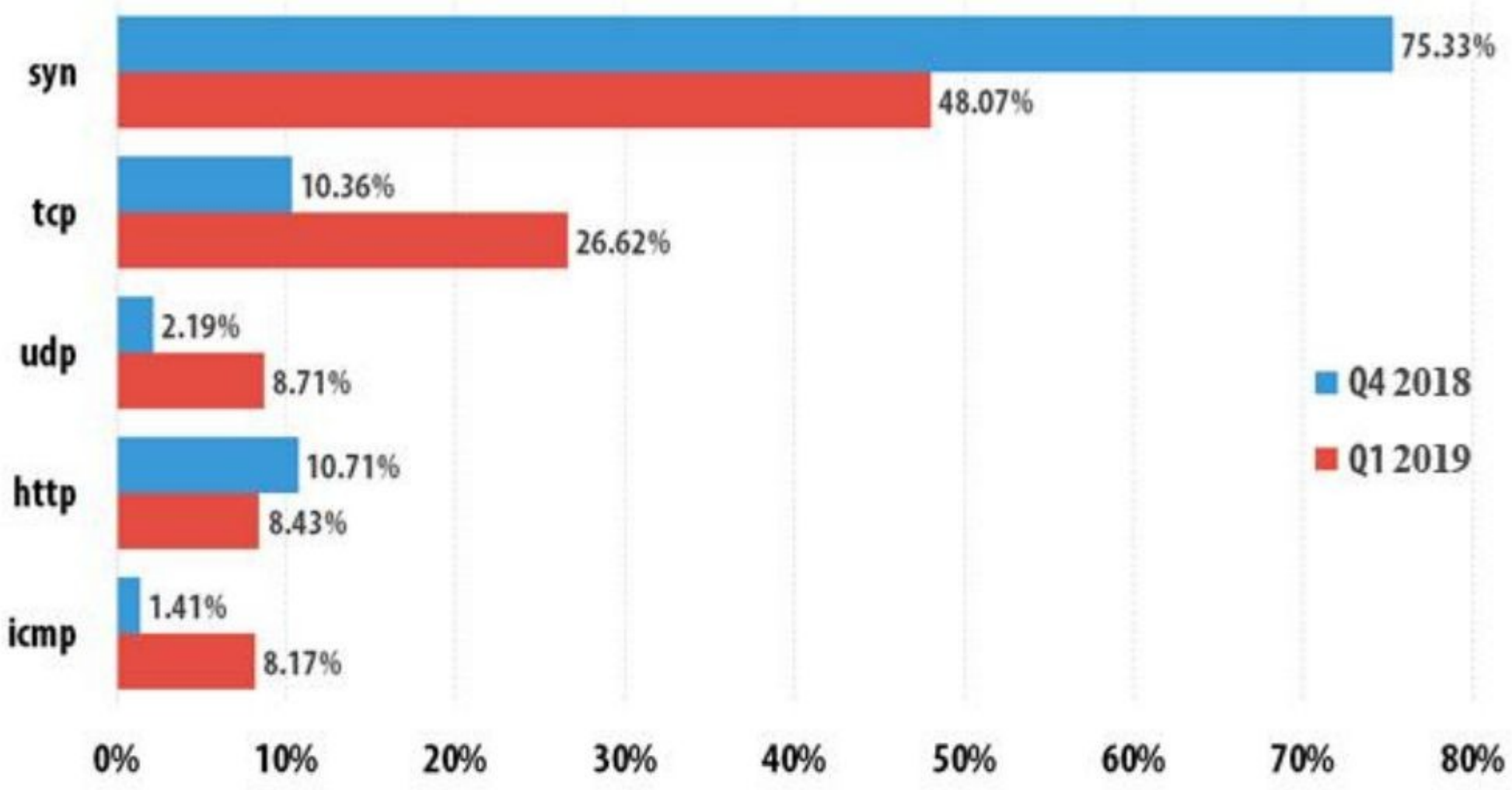
					08-20.МКР.007.00.000 ІЧ4			
Змн	*Арк.*	*№ докум.*	*Підпис*	*Дат*				
Розроб.		*Ковальов В.А.*			*Метод виявлення DDoS-атак.* Схема послідовності взаємодії програмного засобу	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*Дудатьєв А.В.*					1	1
Реценз.		*КрупельницькийЛ.*				*ВНТУ, гр.1БС-18м*		
Н. Контр.		*Дудатьєв А.В.*						
Затверд.		*Лужецький В.А.*						

СХЕМА ЛАНЦЮЖКА ВІДПОВІДАЛЬНОСТЕЙ



					08-20.МКР.007.00.000 ІЧ3			
Змн	*Арк.*	*№ докум.*	*Підпис*	*Дат*				
Розроб.		*Ковальов В.А.*			*Метод виявлення DDoS-атак.* Схема ланцюжка відповідальностей	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*Дудатьєв А.В.*					1	1
Реценз.		*КрупельницькийЛ.*				*ВНТУ, гр.1БС-18м*		
Н. Контр.		*Дудатьєв А.В.*						
Затверд.		*Лужецький В.А.*						

СТАТИСТИЧНА ДІАГРАМА DDOS-АТАК ЗА ВИДАМИ



					08-20.БДР.007.00.000 ІЧ2			
Змн	*Арк.*	*№ докум.*	*Підпис*	*Дат*				
Розроб.		*Ковальов В.А.*			*Метод виявлення DDoS-атак..* Статистична діаграма DDoS-атак за видами	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*Дудатьєв А.В.*					1	1
Реценз.		*КрупельницькийЛ.*				*ВНТУ, гр.1БС-18м*		
Н. Контр.		*Дудатьєв А.В.*						
Затверд.		*Лужецький В.А.*						

ТАБЛИЦЯ ПОРІВНЯННЯ РОЗРОБЛЕНОГО МЕТОДУ ВИЯВЛЕННЯ З АНАЛОГАМИ

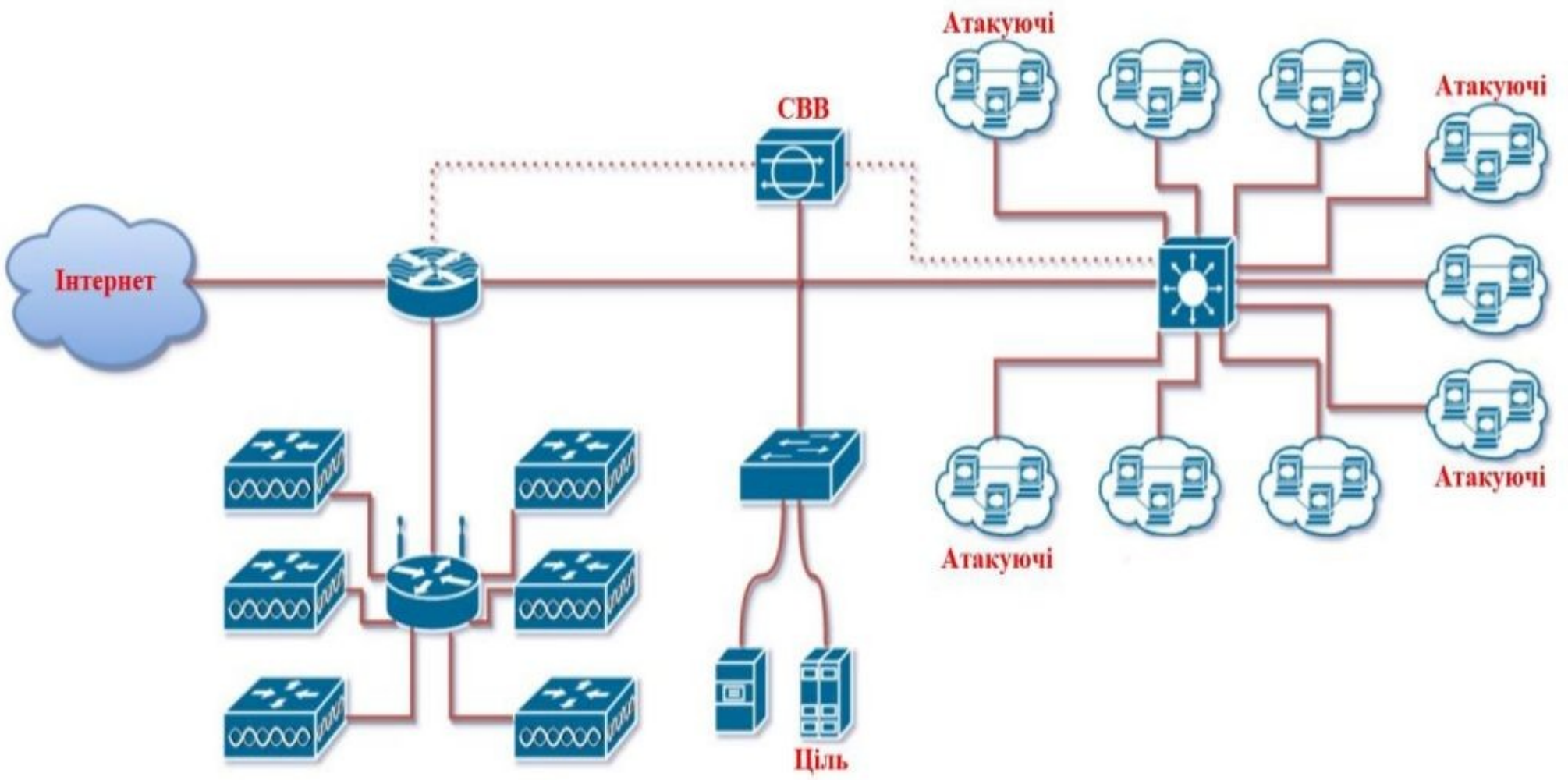
Метод виявлення атаки	Відсоток виявлення атаки	Хибна тривога	Можливість виявляти UDP/ICMP/HTTP атаки
CUSUM-SYN	100%	0%	Hi
CUSUM Ентропія	100%	14%	Так
4EKC Ентропія	100%	0%	Так

					08-20.БДР.007.00.000 ІЧ6
Змн	*Арк.*	*№ докум.*	*Підпис*	*Дат*	
Розроб.		*Ковальов В.А.*			*Метод виявлення DDoS-атак.* Таблиця порівняння розробленого методу виявлення з аналогами
Перевір.		*Дудатьєв А.В.*			
Реценз.		*КрупельницькийЛ.*			
Н. Контр.		*Дудатьєв А.В.*			
Затверд.		*Лужецький В. А.*			

Літ.	*Арк.*	*Аркушів*
	1	1

ВНТУ, гр.1БС-18м

СХЕМА ТЕСТОВОЇ МЕРЕЖІ



					08-20.БДР.007.00.000 ІЧ1			
Змн	*Арк.*	*№ докум.*	*Підпис*	*Дат*				
Розроб.		*Ковальов В.А.*			*Метод виявлення DDoS-атак..* Схема тестової мережі	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*Дудатьєв А.В.*					1	1
Реценз.		*КрупельницькийЛ.*				*ВНТУ, гр.1БС-18м*		
Н. Контр.		*Дудатьєв А.В.*						
Затверд.		*Лужецький В. А.*						

СХЕМА АЛГОРИТМУ РОБОТИ 4ЕКС