Міністерство освіти і науки України
«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____ М.М. Браіловський
“_____”_____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
випускної кваліфікаційної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

напрям підготовки 125 «Кібербезпека»
(код і назва напряму підготовки)

кваліфікація 3439 «Фахівець із захисту інформації в інформаційних і комунікаційних системах»
(код і назва кваліфікації)

на тему: Підвищення механізму виявлення спаму за рахунок застосування алгоритмів розпізнавання кібератак

Виконавець: студент II курсу, групи КБМ-21

_____ **Войтенко Іван Юрійович**
(підпис) (прізвище ім’я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник роботи	Наконечний В. С.	

Нормоконтроль	Фесенко А. О.	

Київ

2021

Міністерство освіти і науки України
«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри
кібербезпеки та захисту інформації

_____ М.М. Браіловський

«_____»_____20___року

ЗАВДАННЯ
на виконання дипломної роботи

Напрям підготовки 125 «Кібербезпека»
(код і назва напряму підготовки)

студенту **КБМ-21** **Войтенку Івану Юрійовичу**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Вдосконалення системи ідентифікації мережевих атак

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол №2 від 11.10.2018 р.

2. ВИХІДНІ ДАНІ ДО РОБОТИ

механізми ідентифікації мережевих атак, алгоритми ідентифікації мережевих атак

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ (перелік питань, що їх належить розробити)

дослідження методів реалізації сучасних мережевих атак включаючи шкідливі листи
дослідження механізмів ідентифікації мережевих атак
аналіз існуючих алгоритмів розпізнавання мережевих атак
порівняльний аналіз запропонованих алгоритмів ідентифікації мережевих атак

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність вибір найоптимальнішого алгоритму з точки зору безпеки для покращення ідентифікації мережевих атак з боку шкідливих листів

5. ДАТА ВИДАЧІ ЗАВДАННЯ 11 жовтня 2020 року

Завдання видав _____ В.С. Наконечний
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ І.Ю.Войтенко
(підпис) (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів випускної кваліфікаційної роботи	Термін виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	11.10.2020-27.10.2020	*виконано*
2	Аналіз літератури	27.10.2020-28.11.2020	*виконано*
3	Аналіз міжнародного документу для системи ідентифікації мережевих атак	28.11.2020-10.12.2020	*виконано*
4	Дослідження процесу ідентифікації мережевих атак з боку шкідливих листів	10.12.2020-25.12.2020	*виконано*
5	Дослідження поширених мережевих атак	25.12.2020-13.01.2021	*виконано*
6	Аналіз алгоритмів ідентифікації мережевих атак з боку шкідливих листів	13.01.2021-28.01.2021	*виконано*
7	Дослідження алгоритмів цифрового спектрального аналізу	28.01.2021-22.02.2021	*виконано*
8	Проведення порівняльної характеристики алгоритмів ідентифікації мережевих атак	22.02.2021-16.03.2021	*виконано*
9	Вибір найефективнішого алгоритму для системи ідентифікації мережевих атак	16.03.2021-01.05.2021	*виконано*
10	Формування висновків і рекомендацій для вдосконалення системи ідентифікації мережевих атак з боку шкідливих листів	1.05.2021-18.05.2021	*виконано*
11	Оформлення пояснювальної записки	20.05.2021-31.05.2021	*виконано*
12	Підготовка до захисту дипломної роботи	1.06.2021-3.06.2021	*виконано*

Студент-дипломник _____ І.Ю. Войтенко
(підпис) (ініціали, прізвище)

Керівник випускної кваліфікаційної роботи _____ В.С. Наконечний
(підпис) (ініціали, прізвище)

УДК 044.056.52

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Вдосконалення системи ідентифікації мережевих атак» складається зі списку скорочень, вступу, основної частини, що містить 3 розділи, висновків, списку літератури та джерел. Загальний обсяг роботи – 93 сторінки. Робота містить 19 рисунків. Список використаних джерел включає 87 джерел.

Об’єкт дослідження – процес ідентифікації мережевих атак.

Мета роботи – вдосконалення системи ідентифікації мережевих атак за рахунок визначення ефективного алгоритму ідентифікації.

Предмет дослідження – механізми ідентифікації мережевих атак.

Метод дослідження – аналіз та вибір ефективного алгоритму для підвищення ефективності ідентифікації мережевих атак.

Практична цінність отриманих результатів полягає у виборі найоптимальнішого ефективного з точки зору інформаційної безпеки алгоритму розпізнавання для покращення системи ідентифікації мережевих атак.

Ключові слова: ідентифікація, мережева атака, механізм захисту, міжмережевий екран, система запобігання вторгнень, алгоритм розпізнавання, система виявлення вторгнень, механізм ідентифікації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ASCII – American standard code for information interchange;

ARP – Address Resolution Protocol;

IT – Інформаційні технології;

ADSL – Asymmetric Digital Subscriber Line

APT – Advanced persistent Threat;

АПЗ – Антивірусне програмне забезпечення;

DMZ – Demilitarized Zone;

DOS – Denial of Service;

DDOS – Distributed Denial of Service;

DNS – Domain Name System;

HIPS – Host Intrusion prevention system;

HTTP – Hypertext Transfer Protocol;

ПЗ – Програмне забезпечення;

Iframe – Inline frame;

IP – Internet protocol;

IPSec – Internet protocol security

ICMP – Internet control message protocol;

ІПІ – Імовірність правильної ідентифікації;

ISO – International Organization for Standardization;

ISP – Internet Service Provider

ЦСА – Цифровий спектральний аналіз;

СУІБ – Системи управління інформаційною безпекою;

MITM – Man-in-the-middle;

NIDS – Network Intrusion detection system;

NBA – Network behavior analytic;

NNIDS – Network noir intrusion detection system;

NGIPS – Next Generation Intrusion prevention system;

NLTK – Natural language toolkit

kNN – Алгоритм класифікації;

OWASP – Open Web Application Security Project;

OSI – Open systems interconnection basic reference model;

PCI DSS – Payment Card Industry Data Security Standard;

PHP – Hypertext Preprocessor;

RHT – Rule Hash Table;

RST – Rule Status Table;

RFC – Request for Comments;

ЗПЗ – Зловмисне програмне забезпечення;

SNMP – Simple Network Management Protocol;

SMB – Server message block;

SQL – Structured query language;

SSL – Secure socket layer;

SIEM – Security information and event management;

IMA – Ідентифікація мережевих атак;

STD – Skip Distance Table;

TCP – Transmission Control Protocol;

TZSP – TaZmen Sniffer Protocol;

TLS – Transport layer security;

UDP – User Datagram protocol;

URI – Uniform Resource Identifier;

WIPS – Wireless Intrusion Prevention System;

XML – eXtensible Markup Language;

XPath – XML Path Language;

ВСТУП

З огляду на останні досягнення інформаційних та комунікаційних технологій, а також всебічно зростаючу потребу в мережевих технологій, інформаційна безпека стає дуже гострою проблемою. Одночасно з цим спостерігається широке застосування Інтернету організаціями та підприємствами різної форми власності для обміну даними різного ступеню конфіденційності. Тому надзвичайно важливим є захист інформаційних ресурсів, що є актуальною проблемою сьогодення.

В останні кілька років, атаки у вигляді шкідливого мережевого трафіку з боку шкідливих листів, які зловмисники використовують для поширення небажаної та шкідливої інформації на комп'ютерні мережі постійно зростають. До таких видів листів можуть належати [1;, 2]:

- фішинг;
- листи з шкідливим ПЗ;
- листи вимагачі;.

Тому для захисту від такого виду атак необхідно правильно та своєчасно ідентифікувати шкідливі листи.

На теперішній час існують різні інструменти та механізми, що спрямовані на забезпечення інформаційної безпеки. До таких механізмів відносяться [3;, 4]:

- антивірусне програмне забезпечення;
- антиспам технології;.

Для підприємства пошта електронна є суттєво важливою, оскільки це є звичайний, доступний і зручний метод для комунікацій з клієнтами, а також спосіб маркетингу. Власне, зараз існують десятки поштових сервісів, компанії зазвичай встановлюють власний поштовий сервер. Через вищевказані переваги пошта електронна дуже часто використовується як засіб для розповсюдження «спаму» [5] .

За По даними компанії «Лабораторії Касперського», частина спаму в світовому трафіку пошти у 2019 році складала у середньому 51% [6]. Значно

гострою ця проблема є для власників підприємств, оскільки адреси їх поштових скриньок є відкритими і через деякий час туди надходять тисячі небажаних листів. Зі стрімким ростом та вдосконаленням механізмів для поширення шкідливих листів, спам стає все більш витонченими та важчими у розпізнаванні. Вибір ефективних алгоритмів з точки зору швидкодії та безпеки для системи ідентифікації шкідливих листів є актуальним завданням цієї дипломної роботи.

Відповідно, метою даної роботи є вдосконалення системи ідентифікації шкідливих листів за рахунок визначення найефективнішого алгоритму розпізнавання. Вибір ефективного алгоритму для одного з найважливіших компонентів системи в інфраструктурі обрано в якості практичної частини роботи.

Для виконання досягнення поставленої в роботі мети необхідно вирішення наступних задач роботи були поставлені такі задачі:

- провести дослідження методів реалізації сучасних мережевих атак з боку розповсюдження шкідливих листів;
- дослідити механізми ідентифікації мережевих атак;
- зробити провести аналіз існуючих алгоритмів розпізнавання шкідливих листів;
- здійснити порівняльний аналіз запропонованих алгоритмів для системи ідентифікації мережевих атак з боку розповсюдження шкідливих листів [9];

Отже, в сучасних реаліях необхідним компонентом архітектури є система обробки та надходження листів. Тому спостерігаючи за ситуацією розвитку шкідливих листів, необхідним рішенням є вибір найефективнішого алгоритму для покращення ідентифікації мережевих атак з боку розповсюдження шкідливих листів.

Дані Результати дипломної роботи були апробовані:

- на 5-й Міжнародна науково-практична конференція (IT&I) [85].
- в науково-практичному журналі кібербезпеки “Scientific and practical cyber security journal [86].
- на 8-й Міжнародна науково-практичній конференції (IT&I) [87].

РОЗДІЛ 1
ХАРАКТЕРИСТИКА МЕРЕЖЕВИХ АТАК

1.1 Варіативність мережевих атак

1.1.1 Розвиток мережевих атак

Технології Інтернет змінюють не тільки підходи організацій до ведення бізнесу, але і їх відношення до забезпечення мережевої безпеки. Для того, щоб компанія була спроможна захистити себе від потенційних та реальних мережних загроз вона повинна мати систему, яка буде контролювати цілісність і автентичність інформації та мережевих з'єднань, що проходять через Інтернет, та через внутрішні мережі [5].

Не є таємницею те, що кожна організація має свою конфіденційну інформацію та критичних важливих ресурсів, компрометація яких може завдати великих збитків. З кожним днем ці ресурси є схильними до спеціальних атак, які поступово стають, з одного боку, більше витонченими, а з іншого боку - простими у виконанні [4]. З цього випливає, що питання захисту інформаційних ресурсів, які циркулюють в мережних джерел є актуальною проблемою сьогодення.

Різноманітність та варіативність мережевих атак є дуже великою, тому важливо мати систему, яка буде правильно та ефективно визначати і запобігати їм.

Зі всебічним розвитком інтернет технологій, а також вільного доступу до різноманітних ресурсів зловмисники все більше обмінюються інформацією та створюють нові методи мережевих атак [6].

Ще одним фактором значиться те, що зловмисник раніше мав володіти хорошими навичками кодування, для того щоб створювати і розповсюджувати просте у використанні ПЗ. На сьогодення для отримання доступу до зловмисного інструментарію, потрібно знайти IP адресу необхідного сайту, потім для проведення атаки достатньо натиснути мишкою по кнопці. Тому мережеві атаки з боку зловмисників є актуальною проблемою сьогодення.

1.1.2 Термін мережевої атаки

Мережева атака зазвичай визначається як вторгнення до певної мережевої інфраструктури, де спочатку аналізується мережеве середовище користувача і відбувається збір інформації для того, щоб використовувати існуючі відкриті порти або уразливості. Така інформація може включати вразливості для несанкціонованого доступу до ресурсів користувача [7]. У тих випадках, коли мета атаки полягає тільки в тому, щоб дізнатися і отримати деяку інформацію з окремої системи, при цьому системні ресурси не зазнають змін або відключені якимось чином, тоді йдеться мова про пасивну мережеву атаку.

Активна атака виникає там, де зловмисник здійснює доступ і змінює, вимикає або знищує певні ресурси або дані. Атака може виконуватися або ззовні організації неуповноваженим суб'єктом (Outside Attack), або з боку компанії «інсайдером», який вже має певний доступ до мережі (Inside Attack). Дуже часто мережева атака поєднується з введенням шкідливих компонентів до цільових систем таке, як зловмисне програмне забезпечення (ЗПЗ) [8].

На даний момент виділяють наступні види мережевих атак [23]:

- mailbombing, переповнення буферу (buffer overflow);
- застосування спеціалізованого ПЗ для зараження або прослуховування мережі;
- мережеве сканування (network scanning);
- підміна IP-адреси (IP-spoofing);
- атаки типу людини посередині (man in the middle);
- ін'єкцїЇ (SQL-ін'єкція, PHP-ін'єкція та інші);
- відмова в обслуговуванні (DoS та DDoS атаки);
- фішингові атаки, тощо.

1.1.3 Фішингові повідомлення, як неавтоматизована ланка проведення атак

З усіх стандартів інформаційної безпеки можна впевнитись у важливості проведення атак на основі соціальної інженерії та фішингу. Стандарти OSSTMM, так і NIST пропонують під час тестування проводити фішингові атаки та атаки на основі соціальної інженерії.

Варто переглянути, що саме зазвичай тестується під час атак [14]:

- Соціальна інженерія — метод перевірки з підключенням «людського фактору», спроможність чітко виявляти і отримувати конфіденційні дані та іншу інформацію шляхом Інтернету чи телефону (співробітники організації чи будь-які інші уповноважені особи, які присутні в мережі організації) [9];.
- Веб-застосунок — застосовується для знаходження вразливості в безпеці та інших проблем в декількох варіантах веб-застосунків і серверів, які розміщені на боці сервера чи клієнта [9];
- Мережева служба. — перевірка мережі для виявлення можливості доступу зловмисників чи інших неавторизованих об'єктів [10];
- Клієнтська частина — для перевірки застосунків, встановлених на клієнтському сайті [10];
- Віддалене підключення — перевірка VPN чи аналогічного об'єкта, який може забезпечити доступ до підключеної системи [11];
- Бездротові мережі — перевірка для бездротових застосунків і сервісів, включаючи різні компоненти та функції [11];
- Перевірка системи автоматичного контролю та збору інформації (SCADA Pentesting). [12].

У всіх сферах, окрім соціальної інженерії пентестеру пропонується широкий вибір рішень, які автоматизують тестування – наприклад, OpenVAS для сканування вразливостей веб-застосунків [18]. Тим часом, у сфері соціальної інженерії автоматизаційних рішень немає. Наразі пентестери мають власноруч вивчати

ситуацію на підприємстві, власноруч добувати інформацію про ціль соціально-інженерної атаки, та власноруч писати фішингове повідомлення адресанту [13].

Сучасна розсилка спаму розповсюджується в тисячах примірників лише за декілька хвилин. Найчастіше всього спам іде через заражені шкідливими ПЗ кінцевими вузлами користувачів.

Що буде в противагу даному методу? Сучасна індустрія інформаційної безпеки надає кількість рішень, і в інвентарі боротьби зі спамом є різніманітні технології. Проте жодна з існуючих технологій не є цілковитим захистом проти даного виду атаки. Панацеї від всіх атак не існує. Велика кількість сучасних рішень застосовують гібридні технологій, бо ефективність продукту не буде висока [15].

1.2 Різноманітність мережевих атак

1. Mailbombing.

Є одним з найвідоміших та найстаріших методів атак. Реалізовується такий метод достатньо простим чином. На поштову скриньку приходить велика кількість поштових повідомлень в результаті чого унеможливлюється робота з цією самою скринькою або взагалі з поштовими серверами. Для реалізації такого типу атаки вже створені спеціалізовані програми в, яких навіть людина без спеціальних знань зможе застосувати такий метод атаки. Для цього потрібно лише ввести e-mail, кількість необхідних повідомлень та текст повідомлення. До таких програм належать: “Mail Bomber”, “Spam Revenge”, “PTF Framework” та інші [15]. Програми для атаки mailbombing зазвичай, приховують реальну IP адресу відправника, застосовуючи для надсилання прихованого поштового серверу. Такий тип мережевої атаки важко запобігти, оскільки поштові системи захисту провайдерів не спроможні чітко позначити легітимного відправника шкідливих листів. Один з варіантів тоді, коли провайдер може обмежити кількість листів одного відправника, але адреса відправника буде створюватись випадковим чином.

2. Мережева розвідка.

В результаті такого типу атаки хакер не здійснює жодних шкідливих дій, проте як результат він може отримати відомості про архітектуру мережі та принципи функціонування обчислювальної системи жертви. Інформація може бути використана для детальної підготовки та підготовки майбутньої атаки.

Під час такої розвідки зловмисник може проводити сканування портів, розвідку операційних систем, які використовуються в певній обчислювальній системі, розвідку адрес, розвідку запитів та серверів DNS [16]. При закінченні розвідки є велика ймовірність, що отримана інформація буде містити DNS-сервери, комп'ютери користувачів, мережеве обладнання, архітектура мережі, сервіси, які запущені на серверах, рівень доступу до цих сервісів для зовнішніх і внутрішніх користувачів.

3. Ін'єкція коду.

Такі методи, об'єднані одним загальним принципом - в результаті атаки дані виконуються як код.

PHP-ін'єкція.

PHP-ін'єкція - це один із методів пошкодження веб-сайту, що побудовані на скриптовій мові PHP. Спосіб полягає у впровадженні спеціально сформованого зловмисного сценарію в код веб-додатку на серверній стороні сайту, що призводить до виконання необхідних команд атакуючому.

Є відомим те, що в більшості поширених безкоштовних пошукових двигунах побудовані форуми, які працюють на мові PHP. Зазвичай такі сайти або форуми мають застарілі версії таких пошукових двигунів в яких є вразливі модулі або окремі конструкції сторінки. Зловмисники аналізують такі вразливості та експлуатують їх у своїх цілях. Прикладом може бути стара вразливість форуму "ExBB", де використовувався зловмисниками окремий запис, а саме:

"GET/mdles/thradstaer/shop.php?odd_data[dir_path]=badsite.com/scrip.txt." [9].

4. Відмова в обслуговуванні.

Атака типу відмова в обслуговуванні перевантажує ресурси системи так, що вона не може надавати відповіді запитам. Атака DDoS також є атакою на ресурси системи, але вона запускається з великої кількості інших вузлів, інфікованих шкідливим ПЗ, які контролюються зловмисником.

На відміну від атак, які дозволяють зловмисникам отримати доступ, відмова в обслуговуванні не забезпечує прямих переваг зловмисникам. Однак, якщо атакований ресурс належить бізнес-конкуренту, тоді користь для зловмисника може бути достатньо реальною. Іншою метою атаки DoS може бути переведення системи в автономний режим, щоб можна було запустити інший вид атаки.

DoS – має за мету перешкодити серверу надавати відповідь на запити. Такий метод атаки не викраде секретну інформації, але зможе допомогти в підготовці інших атак [17]. Наприклад, певне ПЗ через певні помилки в коді може спричинити виключні ситуації і при вимкненні сервісів здатні виконувати код, наданий хакером, або мережеву атаку лавинного типу, коли сервер не зрозумів величезна кількість вхідних пакетів.

5. Розподілена відмова в обслуговуванні.

Підтип DoS атаки, який має ту саму мету, що і DoS атаки, проте проводяться не з одного вузла, а з деяких. В таких методах застосовується появлення помилок, які призводять до відмови служби, яка може також призвести до блокування роботи сервісу, і в результаті до відмови в обслуговуванні.

DDoS застосовується там, де звичайний DoS не є ефективним. Для проведення такої атаки декілька вузлів об'єднуються, і кожний генерує трафік DoS на вузол жертви. Це має назву DDoS. Розмір ботнет мережі може становити від декількох десятків до декількох сотень тисяч вузлів.

Будь-яка спроба проведення атаки має на меті застосування вразливості в системі безпеки жертви. Також даний вид атаки застосовується для спричинення шкоди вузлу. Причиною будь якої експлуатації атаки є майстерність атакуючих, і

недостатня компетентність адміністраторів безпеки, вади з ПЗ і мала увага до інформаційної безпеки в компанії в цілому.

Існують різні типи атак DoS і DDoS. Найбільш поширеними є атака TCP SYN, атака teardrop, атака smurf, атака ping-of-death та атака за допомогою ботнетів [18].

Атака потоку TCP SYN. У цій атаці зловмисник використовує застосовує використання буферного простору під час трьохстороннього рукостискання для ініціалізації протоколу керування передачею (TCP). Пристрій зловмисника переповнює чергу в робочій системі запитів на підключення, але не закінчує встановлення сеансу, коли цільова система відповідає на ці запити. Це призводить до того, що цільова система очікує відповідь від пристрою зловмисника, що призводить до збою системи або стає непридатною для використання, коли черга для підключення заповнюється.

Контрзаходи для атаки з потоком TCP SYN [23]:

- розташування серверів за брандмауером, налаштованим для зупинки вхідних пакетів з прапором SYN;
- збільшити розмір черги з'єднання та зменшити час очікування для відкритих з'єднань.

Атака Teardrop. Ця атака призводить до зміщення полів довжини і фрагментації в послідовних пакетах. В результаті чого пакети перекривають один одного на атакованому хосту. Атакована система намагається дефрагментувати пакети під час процесу, але оскільки вони перемішані це не вдається. В результаті тривалих обчислювальних операцій цільова система не витримує навантаження та стає непрацездатною.

Якщо користувачі не мають виправлень для захисту від цієї DoS-атаки, необхідно вимкнути протокол SMBv2 і заблокувати порти 139 і 445 [12].

Атака типу Smurf. Ця атака передбачає використання IP-спуфінгу та ICMP, щоб надіслати велику кількість пакетів на цільову мережу трафіком. Цей метод атаки використовує запити ICMP, націлені на певні IP-адреси. Ці Такі запити ICMP надпоходять від підробленої адреси жертви. Наприклад, якщо адресою жертви є

10.0.0.10, зловмисник підмінить запит ICMP від 10.0.0.10 до широкомовної адреси 10.255.255.255 [19]. Цей запит буде спрямований на всі IP-адреси в діапазоні, причому всі відповіді повертаються до 10.0.0.10, перевантажуючи мережу. Цей процес є повторюваним і може бути автоматизованим для створення величезної кількості перевантажень в мережі.

Для захисту своїх пристроїв від цієї атаки, потрібно вимкнути трансляцію IP-адрес на маршрутизаторах. Це дозволить запобігти запиту широкомовлення ICMP на мережевих пристроях [9]. Іншим варіантом було б налаштування на кінцевих системах, щоб уникнути їхнього реагування на ICMP-пакети з широкомовних адрес [19].

Атака типу Ping of Death. Цей тип атаки використовує IP-пакети для навантаження цільової системи мережевими пакетами з розміром IP, що перевищує максимум 65535 байт. IP-пакети такого розміру не допускаються, тому зловмисник фрагментує IP-пакет. Після того, як цільова система знову збирає пакет, вона може бути вразлива до переповнення буфера та інших збоїв.

Атака типу “Ping of death” може бути заблокована за допомогою брандмауера, який перевіряє фрагментовані IP-пакети для максимального розміру [21].

Для вирішення цієї проблеми створено повноваження сертифікатів і хеш-функції. Коли особа 2 (P2) хоче надіслати повідомлення P, і P хоче бути впевненим, що A не буде читати або змінювати повідомлення і що повідомлення дійсно надходило з P2, повинен бути використаний наступний метод. На рисунках 1.1 та 1.2 показана реалізації такого типу атаки [7]:

На рисунках 1.1 та 1.2 зображено:

- Перший етап прослуховування сесії;
- Етап, коли сесія прослуховується.
- Етап, при перехоплення сесії трафіку.

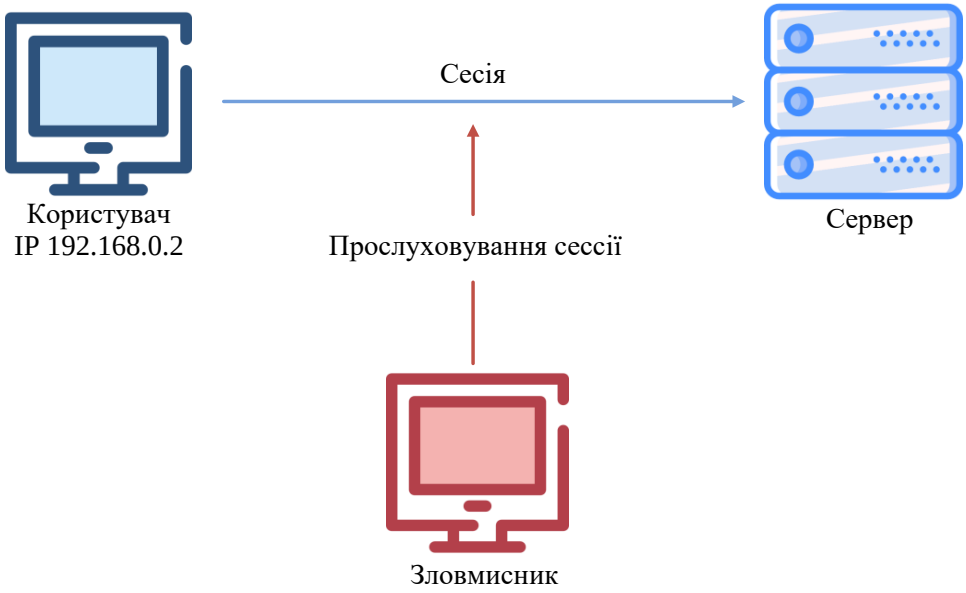

Рисунок 1.1 – Прослуховування сесії

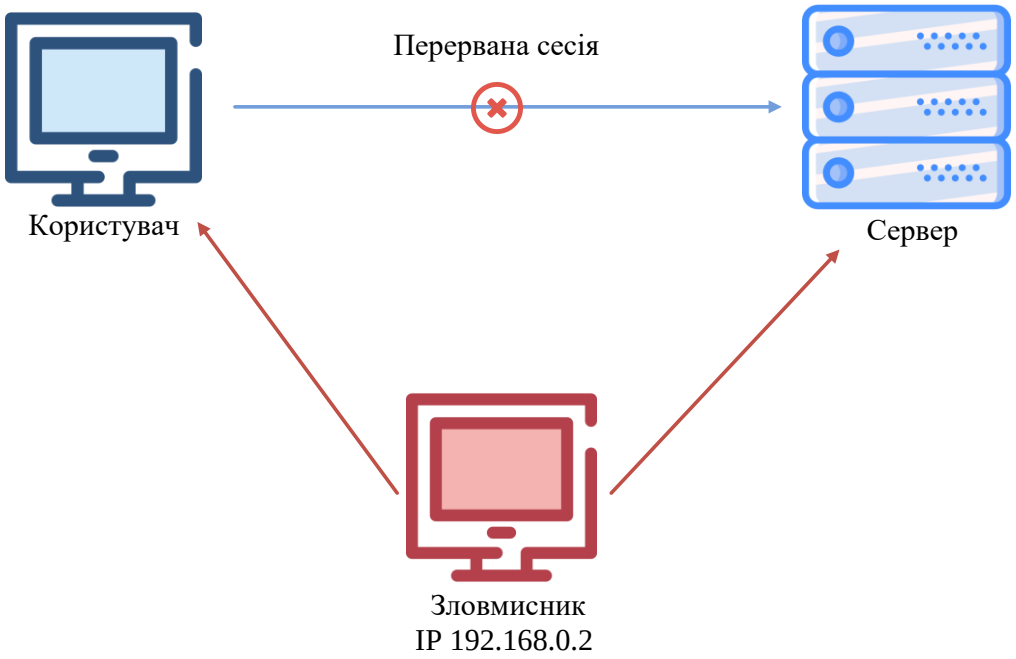

Рисунок 1.2 – Перехоплення сесії

- P2 створює симетричний ключ і шифрує його за допомогою відкритого ключа P.
- P2 посилає зашифрований симетричний ключ P.
- P2 обчислює хеш-функцію повідомлення і цифрово підписує його.

- P2 шифрує своє повідомлення і хеш з підписом повідомлення, використовуючи симетричний ключ і передає всю річ P.

Таким чином зловмисник перехопив контроль над сеансом та видав свій комп'ютер за локальний комп'ютер користувача.

- P здатний приймати симетричний ключ від P2, оскільки тільки він має приватний ключ для розшифрування шифрування.
- P, і тільки P, може розшифрувати симетрично зашифроване повідомлення і підписаний хеш, оскільки він має симетричний ключ. Він може перевірити, що повідомлення не було змінено, оскільки може обчислити хеш отриманого повідомлення та порівняти його з цифровим підписом.
- P також в змозі довести собі, що P2 був відправником, оскільки тільки P2 може підписати хеш так, що він перевіряється за допомогою відкритого ключа P2.

6. Атака типу Replay.

Атака відтворення відбувається, коли зловмисник перехоплює та зберігає старі повідомлення, а потім намагається відправити їх пізніше, видаючи себе за одного з учасників. Цей тип атаки можна легко протиставити мітками сеансів (випадкове число або рядок, який змінюється з часом).

7. Соціальна інженерія.

Соціальна інженерія – під цим терміном розуміють застосування непрофесіоналізму. некомпетентності, або халатності персоналу для отримання доступу до певної інформації [24]. Такий спосіб часто проводиться без робочої станції, із застосуванням телефону або комунікацій між людьми. За допомогою такого методу зловмисник може розвідати достатньо цінну інформацію. В процесі атаки атакуючий вводить в оману свою жертву та після цього, отримує відомості, які важко отримати вибравши інший метод.

8. Фішинг і націлений фішинг.

Фішинг-атака - це практика надсилання повідомлень електронної пошти, які надсилаються нібито з надійних джерел, з метою отримання особистої інформації або впливу на користувачів [27]. Вона поєднує в собі соціальну інженерію та

технічні хитрощі. Це може включати додаток до електронного листа, який завантажує шкідливі програми на комп'ютер. Це також може бути посилання на незаконний веб-сайт, який може обманювати завантаженням шкідливих програм або передачею особистої інформації.

Націлений фішинг є направленою атакою на окремого користувача. Зловмисники не поспішають проводити дослідження цілей та створювати особисті та відповідні повідомлення. Через це, Націлений фішинг дуже важко виявити. Одним з найпростіших способів, яким зловмисник може провести атаку Націлений фішингу, є підробка електронною поштою, тобто, коли інформація в розділі "Відправник" електронної пошти є фальсифікованою, що робить її такою, якою б вона надійшла від знайомого, наприклад вашого керівництва або компанії-партнера. Іншим методом, який використовують шахраї, щоб додати довіру до своєї історії, є клонування веб-сайту - вони копіюють легітимні веб-сайти, щоб обдурити користувача введенням особистої інформації або облікових даних для входу.

Щоб зменшити ризик фішингу, можна використовувати такі методи [13]:

- Критичне мислення – не сприймати, що повідомлення в електронній пошті є справжніми. Необхідно аналізувати кожний невідомий лист.
- Наведення курсору на посилання - навести курсор миші на посилання, але не натискати на нього. Через пару секунд над курсором з’явиться посилання на URL-адресу. Виходячи з неї можна обміркувати чи варто на нього переходити.
- Аналіз заголовків електронної пошти - заголовки електронної пошти визначають спосіб отримання електронної пошти на вашу адресу. Параметри «Відповісти» повинен вести до того ж домену, який вказаний у повідомленні електронної пошти.
- Пісочниці – існує технологія перевірки вмісту електронної пошти в середовищі пісочниці. Спочатку лист проходить спеціальний аналіз в пісочниці і тільки потім “чистий“ лист потрапляє до користувача.

9. Атаки з використанням шкідливих програм.

Такий вид атаки реалізується, коли на комп’ютер жертви спеціально або випадково встановлюється програма, яка містить в собі вірус. За своїм типом, вірус може нести деструктивні дії комп’ютеру жертви, поширюватися між комп’ютерами однієї мережі, створення прихованого каналу зв’язку з власником вірусу і передачі конфіденційної інформації та інші.

Зловмисний код можна назвати шкідливим ПЗ, яке встановлено у системі без згоди користувача. Таке, програмне забезпечення може приєднуватися до легітимного коду і поширюватися, ховатися в корисних програмах або реплікувати себе через Інтернет. Ось деякі з найбільш поширених типів шкідливих програм:

- Макровіруси – такий вид вірусів заражають програми, такі як Microsoft Word або Excel. Макровіруси додаються до послідовності ініціалізації програми. Після відкриття програми вірус виконує інструкції перед передачею управління до програми. Вірус копіюється і приєднується до іншого коду в комп'ютерній системі[14].

- Файлові інфектори - віруси файлових інфекторів зазвичай приєднуються до виконуваного коду, наприклад файлів “.exe”. Вірус встановлюється під час завантаження коду. Інша версія файлового інфектора асоціюється з файлом, створюючи файл вірусів з тим же ім'ям, але розширення “exe”. Тому, коли файл відкривається, код вірусу буде виконуватися [20].

- Системні або завантажувальні інфектори - вірус запису завантаження підключається до головного завантажувального запису на жорстких дисках. Коли система запускається, вона буде дивитися на завантажувальний сектор і завантажувати вірус в пам'ять, де він може поширюватися на інші диски і комп'ютери [17].

- Поліморфні віруси - ці віруси приховують себе за допомогою різних циклів шифрування і дешифрування. Зашифрований вірус і пов'язаний з ним механізм мутації спочатку розшифровуються за допомогою програми дешифрування. Вірус продовжує заражати область коду. Функціонал мутації потім розробляє нову процедуру дешифрування, і вірус шифрує функціонал мутації і копію вірусу з

алгоритмом, що відповідає новій процедурі дешифрування. Зашифрований пакет функціонал мутації і вірусу додається до нового коду, і процес повторюється. Такі віруси важко виявити, але мають високий рівень ентропії через багато модифікацій їх вихідного коду. Антивірусне програмне забезпечення або засоби моніторингу можуть їх виявити [24].

- Приховані віруси – такий вид вірусів бере на себе функції системи, щоб приховати себе. Вони роблять це шляхом компрометації програмного забезпечення для виявлення шкідливого ПЗ, завдяки чому повідомляється, що інфікована область не інфікована. Ці віруси приховують будь-яке збільшення розміру зараженого файлу або зміни до дати та часу файлу останньої модифікації [27].
- Троянський вірус - це програма, яка ховається у легітимному додатку або програмі і зазвичай має зловмисну функцію. Основна відмінність між вірусами і троянами полягає в тому, що трояни не самостійно реплікуються. На додаток до запуску атак на систему, троян може встановити бекдор, який можуть використовувати зловмисники. Наприклад, троян може бути запрограмований на відкриття окремого порту, щоб зловмисник міг використовувати його для прослуховування і виконання атаки [21].
- Логічні бомби - логічна бомба є типом шкідливого програмного забезпечення, яке додається до програми і викликається певним випадком, наприклад, логічним умовою або конкретною датою та часом [28].
- Дропери - це програма, яка використовується для встановлення вірусів на комп'ютерах. У багатьох випадках дропери не інфікована шкідливим кодом і тому не може бути виявлена програмним забезпеченням для вірусів. Дропер також може підключатися до Інтернету та завантажувати оновлення до вірусного програмного забезпечення, що перебуває у порушеній системі [28].
- Вірус вимагач - це тип шкідливого програмного забезпечення, який блокує доступ до даних жертви і загрожує його публікацією або видаленням, якщо не буде сплачено викуп. Зазвичай, такий тип вірусів шифрує дані і для отримання ключа дешифрування користувачеві необхідно заплатити викуп.

Висновки до розділу 1

У розділі 1 було проаналізовано характеристику та види мережевих атак з боку шкідливих листів. Таким чином можна зробити висновок, що з всебічним розвитком інтернет технологій зловмисникам все легше обмінюватись інформацією, створювати та вдосконалювати види поширення шкідливих листів.

Для поширення спам листів зловмисники використовують різні підходи, а саме:

- фішинг;
- антирекламу;
- листи з шкідливим вмістом;
- листи вимагачі;.

З кожним днем механізми проведення атак покращуються і тому важливо правильно розпізнати певний вид атаки. Для ефективного захисту проти різноманітних шкідливих листів необхідне їх ґрунтовне розуміння та реалізація. Як описано вище мережеві атаки направлені на різні вектори системи і складно передбачити те, куди буде спрямована атака.

При успішній атаці зловмисники можуть викрасти певні конфіденційні дані користувача або компанії і в подальшому монетизувати ці дані. Націлені фішингові компанії та соціальну інженерію не можна з повною впевненістю віднести до мережевих атак, проте їх поширення та реалізація відбувається безпосередньо через мережу, тому такий вид атаки описаний вище.

Заходи які, спрямовані на запобігання цих загроз, змінюються, але основи безпеки залишаються незмінними. До них відносяться: оновлення систем та антивірусних баз, навчання працівників, налаштування брандмауера на білі списки лише потрібних портів і вузлів, збереження надійності паролів, використання моделі з найменшими привілеями в IT-середовищі, регулярні резервні копії та безперервна перевірка IT-систем на підозрілу діяльність в мережі за допомогою систем ідентифікації мережевих атак з боку шкідливих листів.

РОЗДІЛ 2
МЕХАНІЗМИ РОЗПІЗНАВАННЯ ШКІДЛИВИХ ЛИСТІВ

2.1 Поширені механізми ідентифікації мережевих атак

2.1.1 Антивірусне програмне забезпечення

Антивірусне програмне забезпечення (АПЗ) - це комп'ютерна програма, яка використовується для запобігання, виявлення та видалення шкідливих програм [31].

Антивірусне програмне забезпечення було спочатку розроблено для виявлення та видалення комп'ютерних вірусів. Проте, з поширенням інших видів шкідливого програмного забезпечення, АПЗ почало надавати захист від інших комп'ютерних загроз. Зокрема, сучасне АПЗ може захистити користувачів від: шкідливих допоміжних об'єктів браузера, викрадачів даних у браузерів, вірусів вимагачів, клавіатурних шпигунів, бекдорів, руткітів, троянських програм, черв'яків, програм закладок, шахрайських програм, рекламних програм і шпигунських програм.

Деякі продукти також включають захист від інших комп'ютерних загроз, таких як інфіковані та шкідливі URL-адреси, спам, афери та фішинг атаки, онлайнову ідентифікацію, атаки онлайн-банкінгу, методи соціальної інженерії, націлені атаки та атаки за допомогою ботнет мереж, як приклад DDoS [22].

Антивірусні інструменти є надзвичайно важливими для користувачів, які мають встановлювати певні програми та оновлюватись, оскільки комп'ютер без захисту від вірусів буде заражений протягом декількох хвилин після підключення до Інтернету.

Шкідливі програми створюються щодня, а це означає, що антивірусним компаніям доводиться регулярно оновлювати свої засоби виявлення, щоб мати справу з більш ніж 60 000 нових видами шкідливих програм [35].

Шкідлива програма на сьогоднішній день може швидко змінюватись, тим самим роблячи її виявлення складнішим. Віруси можуть бути запрограмовані на

пошкодження даних користувача, запобігання доступу користувача до даних або керування комп'ютером.

Антивірусні програми виконують ряд функцій, а саме [22]:

- Сканувати певних файлів або каталогів на наявність будь-яких шкідливих програм або відомих шкідливих шаблонів;
- Дозволяє планувати сканування для автоматичного запуску;
- Дозволяє в будь який час ініціювати сканування певного файлу, всього комп'ютера, компакт-диска або флеш-пам'яті
- Видаляти будь-який виявлений шкідливий код
- Показувати статус комп'ютера.

2.1.2 Пісочниці

Пісочниця - це середовище тестування, яке ізолює неперевірені зміни коду та безпосереднє експериментування від виробничого середовища або сховища, в контексті розробки програмного забезпечення, включаючи керування веб-розробкою та переглядом.

Технологія пісочниці, яка зображена на рисунку 2.1, захищає сервери та їхні дані, перевірені дистрибутиви вихідного коду та інші колекції коду, даних та вмісту, які локальні або загальнодоступні, від змін, які можуть завдати шкоди. Пісочниці копіюють мінімальну функціональність роботи користувача за комп’ютером, необхідну для точного тестування програм або іншого розроблюваного коду [39].

Концепція пісочниці зазвичай вбудовується в програмне забезпечення для контролю перегляду, де розробники перевіряють копію дерева вихідного коду. Тільки після того, як розробник повністю протестував зміни коду у власній пісочниці, зміни будуть перевірені та об'єднані зі сховищем і тим самим будуть доступні іншим розробникам або кінцевим користувачам програмного забезпечення [23].

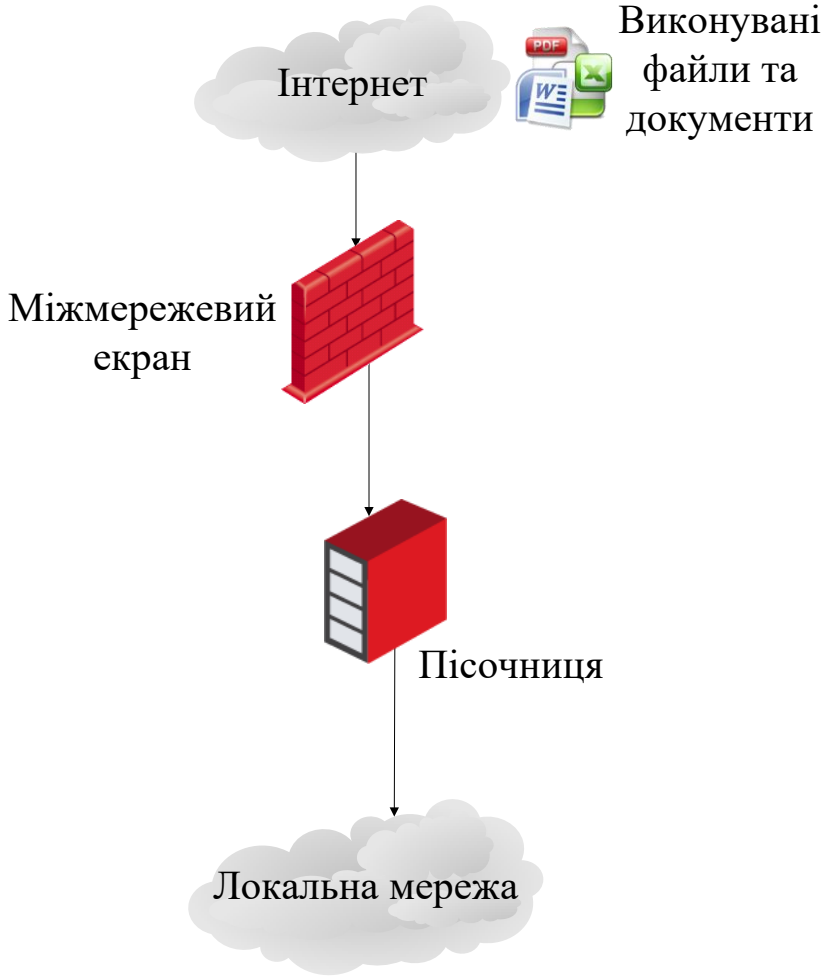

Рисунок 2.1 – Робота пісочниці

Таким чином відбувається очищення файлів від вірусів, які містяться в них.

2.2 Еволюція спаму та антиспамерських програм

Сьогодні існує кілька способів запобігти спаму. Одним із методів є превентивний захист, багато з яких полягають у тому, щоб не дозволяти спамерам виявляти адреси електронної пошти, наприклад, не поширювати адреси на загальнодоступних сайтах або час від часу змінювати їх електронні адреси. Звичайно, для власників бізнесу цей варіант неприйнятний, оскільки користування послугами компанії призведе до того, що клієнти почуватимуться некомфортно. Тому було розроблено кілька типів фільтрації електронної пошти.

Всього за десятки хвилин сучасний спам поширив тисячі копій. Найпоширенішим є те, що спам проходить через комп'ютер користувача, зараженого шкідливим програмним забезпеченням - ботнетом. Чому цьому тиску можна протистояти? Сучасна індустрія інформаційної безпеки пропонує безліч рішень, а в відправниках антиспаму існує безліч методів. Однак жодна з існуючих технологій не може повністю протистояти спаму. [24].

2.2.1 Прямі розсилки

Спам починається з прямої пошти - спамер надсилає пошту зі свого поштового сервера на своє ім’я. Запобігти цьому типу спаму дуже просто (за адресою поштового сервера або адресою відправника). Як тільки це блокування стане загальним, спамери будуть змушені почати підробляти адреси відправника та іншу технічну інформацію. [25].

2.2.2 Розвиток шкідливих листів

Розробка методів розповсюдження шкідливого листя залежить від розробки фільтруючих інструментів. Після того, як метод розповсюдження прийнятий, з’являється ефективний засіб боротьби, і тим, хто надсилає спам, потрібно змінити свої методи. У той же час, чим більша проблема, тим вищий ентузіазм знаходити контрзаходи та швидший технологічний прогрес спамерів - їхній бізнес продовжує зростати, і їм потрібно більше інвестувати у розвиток [26].

2.2.3 Розсилки через релеї

Open Relay - це поштовий сервер, який дозволяє будь-якому користувачеві надсилати електронну пошту на будь-яку адресу. У середині 90-х усі поштові сервери були відкритими ретрансляторами, тому нам довелося змінити та переналаштувати програмне забезпечення на всіх поштових серверах по всьому

світу. Не всі адміністратори поштової системи виконують цю операцію досить швидко, тому для таких серверів існують служби пошуку, а потім їх списки (включаючи списки реального часу на основі DNS-RBL, списки чорних дір у реальному часі)- RBL, realtime blackhole list) і блокування прийому пошти з таких машин [27].

В сьогоденні цей спосіб поширення застосовується, оскільки відкриті релеї існують і в наш час.

2.2.4 Розсилки з модемних пулів

Як тільки відкрита ретрансляційна пошта стає недійсною, спамер починає використовувати пошту від комутованого з'єднання, яке зазвичай використовує таку функцію: поштовий сервер провайдера отримує пошту від свого клієнта та пересилає її, а комутоване з'єднання отримує його динамічно, тобто кожне нове з'єднання після зміни є IP-адресою, тому спамери можуть надсилати пошту з великої кількості IP-адрес [28].

У відповідь провайдери почали встановлювати обмеження на кількість електронних листів, надісланих одним користувачем. Існує чорний список комутованих адрес та блокування отримання пошти із "зовнішнього" модемного пулу.

2.2.5 Розсилки з proxy-серверів

На початку 2000-х, із поширенням високошвидкісних з'єднань, таких як ADSL, спамери почали використовувати вразливості на клієнтських пристроях. Багато модемів ADSL мають вбудований сервер SOCKS або проксі HTTP (програмне забезпечення, яке дозволяє обмінюватися каналами Інтернету між кількома комп’ютерами), і доступ до них можна отримати з будь-якої точки світу без паролів та контролю доступу (для спрощення налаштувань кінцевих користувачів) .

Тому будь-яку операцію (включаючи спам) можна виконати з IP-адреси користувача ADSL [29].

Оскільки таких користувачів по всьому світу мільйони, проблема частково вирішена лише зусиллями виробників обладнання - відкритий для світу "посередник" більше не входить в обладнання.

2.2.6 Злом вузлів користувацьких

На сьогоднішній день більшість електронних листів надсилаються з комп'ютера користувача. На комп'ютері користувача так чи інакше встановлено програмне забезпечення „Троян" (Троян). Спамер може отримати доступ до комп'ютера користувача без відома та контролю користувача.

Наступні методи можуть бути використані для вторгнення в комп'ютер користувача: троянські програми, що розповсюджуються через піратське програмне забезпечення через мережі обміну файлами (наприклад, rghost або depositfiles), використовуючи вразливості в різних версіях Windows та велике програмне забезпечення, яке встановлює бэкдор-програми на комп'ютері користувача, черв'яки електронної пошти. Це продукт останнього покоління, який також використовується для встановлення задніх дверей.

Сьогодні ці програми дуже складні - вони можуть оновлювати свої версії, отримувати інструкції від заздалегідь підготовлених сайтів або IRC-каналів, надсилати спам та виконувати DDoS-атаки.

Згідно зі статистикою Return Path, 96,7% комп'ютерів, які надсилають електронні листи, контролюються зловмисниками [30].

Поява засобів виявлення спаму, заснованих на аналізі вмісту, призвело до розвитку вмісту спаму - вони написані таким чином, що ускладнює автоматичний аналіз. Як і зміна способу використання електронної пошти, спамери змушені боротися зі спамом.

2.2.7 Прості текстові та HTML-листи

На появі шкідливим листів з повідомленнями, вони були однаковими. Одержувачі отримували схожий текст. Такі типи повідомлень насправді легко фільтрувати (за частотою повторення однотипних символів).

2.2.8 Персоналізовані повідомлення

Наступним кроком є додавання налаштувань персоналізації (наприклад, "привіт, Сергій!" - на початку листа на Serhii@bro.net), що робить всі повідомлення різними. Тепер для їх фільтрації необхідно знайти константний рядок і ввести його до списку правил фільтра. Як метод контролю пропонується метод, який може протистояти незначним змінам тексту, і метод статистичної фільтрації, який можна вивчити, такий як байєсівська фільтрація [31].

2.2.9 Техніка випадкових текстів

Спамери можуть витягувати класичний текст або випадкові набори слів на початку або в кінці листа. Ви можете ввести "невидимий" текст (дуже маленький шрифт або колір, який відповідає кольору тла) у повідомленні HTML. Ці включення ускладнюють роботу нечітких підписів та статистичних методів. У відповідь люди провели пошукові цитати, відмовились додавати текст, детальний аналіз HTML та інші методи поглибленого аналізу змісту листа. У багатьох випадках можна визначити факт використання шкідливих трюків та блокувати такі повідомлення, як спам, не вимагаючи детального аналізу його тексту [32].

2.2.10 Ілюстраційні листи

Рекламні повідомлення можуть надсилатися користувачам у вигляді графічних файлів - це значно збільшить складність автоматичного аналізу. У відповідь є кілька методів аналізу зображень, які можуть витягувати з них текст. На жаль, така технологія є дуже цінною і мало поширеною.

Графічний спам - це різноманітна пошта. Деякі з цих електронних листів зі спамом - це прості картинки, які можна ідентифікувати за допомогою фільтрів спаму. Однак спамери все частіше використовують складні типи графічних букв: вони використовують картинки з галасливим фоном, стрибки літер і ліній (тобто букви нерівні щодо ліній), а окремі літери замінюють на Picture, розширюють зображення на кілька градусів , використовуйте рідкісні шрифти або інші розміри шрифтів і намагайтеся обійти спам-фільтр. Текст на зображенні спаму зазвичай стає майже нечитабельним, через що одержувач не може оцінити пропозицію спаму, а основна мета електронного листа не може бути досягнута, тобто реклама недійсна, тому ці методи не є поширеними серед спамерів [33].

Інший прийом для спамерів - використання анімації. Це не спам, що надсилається у вигляді звичайних статичних "картинок" (графічних вкладень), а у вигляді анімованих графічних зображень. Спамери використовують анімований GIF, оскільки його можуть розпізнати всі сучасні браузери та відтворювати автоматично. Зазвичай анімаційний спам містить від 2 до 4 кадрів, з яких лише один кадр є значущим, тобто містить інформаційну частину.

Спамери регулярно намагаються оновити технологію, яка генерує графічні вкладення до електронної пошти зі спамом. Наприклад, розміщуйте файли зображень на сторінках безкоштовного хостингу зображень, таких як imagehask.us, [34]. Коли одержувач відкриває електронне повідомлення, у більшості популярних поштових клієнтів зображення автоматично завантажується із зазначеної URL-адреси. Той самий метод використовується для вкладень PDF. Цей тип вкладення не буде автоматично відкриватися або завантажуватися. Щоб переглянути спам,

користувачі повинні використовувати Acrobat Reader або подібні інструменти, щоб відкрити вкладення.

2.2.11Перефразування текстів

Те саме рекламне повідомлення містить багато варіантів одного і того ж тексту. Кожен окремий лист виглядає простим і зв'язним текстом, і лише маючи багато копій електронного листа, можна визначити тлумачення факту. Таким чином, ви можете ефективно налаштувати фільтри лише після отримання більшості електронних листів. [35]

Сьогодні останні три методи широко використовуються - не всі засоби боротьби зі спамом можуть обробляти їх належним чином, що дозволяє надавати спам користувачам, які використовують старі інструменти фільтрації.

2.3Технології захисту від спаму

2.3.1Економічний підхід

Зрештою, низька вартість Інтернет-послуг призвела до поширення спаму. Використовуючи електронну пошту, зловмисникам не потрібно турбуватися про щохвилинну оплату та поштову оплату, як і надсилання звичайних листів та телефонні дзвінки. А платячи за доступ до Інтернету лише щомісяця, спамери можуть безкоштовно розповсюджувати мільйони електронних листів.

Тому Девіс вважає, що проблему спаму слід вирішувати економічно [36]. Електронна пошта повинна шкодити спамерам. У цьому випадку закони та штрафи безсилі, оскільки, по-перше, спам зазвичай надсилається ненавмисно, а власник робочої станції може навіть не знати, що він надсилає спам; по-друге, іноді його джерело важко знайти, а по-друге, закон правоохоронні органи До цього питання не завжди слід ставитися з належною обережністю.

Існує рішення щодо спаму, яке називається "зв'язок відправника" або "зв'язок уваги". Суть полягає в тому, що відправник пошти вносить суму, щоб підтвердити, що пошта не є спамом. Якщо одержувач позначить це повідомлення як спам, сума буде перерахована на номінальний рахунок.

Система покладала великі надії і була перевірена IronPort, Vanquish та Microsoft на початку 2000-х. Однак функціонування цього механізму важко зрозуміти, і серед учасників відбулися внутрішні боротьби. В результаті інтерес інвесторів зменшився, а тестування припинилося. У вступній промові на Світовому економічному форумі Білл Гейтс передбачив, що ця технологія закінчить спам. На жаль, його бачення було трохи вперед, що викликало лише насмішки.

2.3.2 Чорні списки

DNSBL (список чорних дір на основі DNS), стара назва RBL (список чорних дір у реальному часі), є однією з найстаріших технологій боротьби зі спамом [35]. Усі електронні листи, надіслані з IP-сервера в цьому списку, заблоковані. Існує також спосіб його реалізації, ви можете створити список комп’ютерів, які можна використовувати для надсилання пошти, і створити список адрес клієнтів, які не можна використовувати як поштові сервери. Найпоширеніша і найпростіша реалізація заснована на використанні DNS, звідси і її назва.

Переваги цієї технології: чорний список на 100% відсікає електронні листи з підозрілих джерел, покращує швидкість роботи, не потребує інсталяції іншого програмного забезпечення на сервері та простий у налаштуванні [38].

Недоліки: Швидкість помилкових позитивних даних висока. Спаму, надісланого одним комп’ютером, достатньо, щоб заблокувати всю мережу. Для деяких списків даних відносно мало, тому не всі нові спамери у списку доступні.

2.3.3 Сірі списки

Впровадження методу сірого списку базується на тому, що "поведінка" програмного забезпечення, призначеного для масового розповсюдження спаму, суттєво відрізняється від поведінки звичайних поштових серверів. Основна відмінність полягає в тому, що, на відміну від вимог протоколу SMTP, спамери, як правило, не надсилають електронні листи, коли є тимчасова помилка або немає відповіді. На початковому етапі всі невідомі клієнту сервери заносяться до сірого списку і листи з них блокуються [39].

Сервер, що надсилає пошту, повертає код помилки часу, тому звичайні електронні листи, які не є спамом, не будуть втрачені, а відправлені лише з невеликою затримкою. Вони залишаться в черзі та будуть відправлені на сторону відправника. Через певний проміжок часу наступного разу повторіть спробу.

Якщо поведінка сервера відповідає стандарту, такий сервер буде автоматично доданий до білого списку, а наступні електронні листи прийматимуться без затримки [40].

Цей метод дозволяє усунути до 90% спаму, і майже немає ризику втратити важливі електронні листи, але зараз ця кількість значно зменшилась. Можливість помилок також велика. Наприклад, електронні листи від серверів, які не повністю відповідають рекомендаціям протоколу SMTP, можуть бути помилково вилучені, наприклад різні електронні листи з новинних сайтів.

Більше того, затримка з повторною розсилкою листів може сягати півгодини, іноді навіть більше, що неприпустимо для термінових листів. Але цей недолік виникає лише під час першого прийому і додатково компенсується тим, що затримка виникає лише тоді, коли перший лист відправляється з невідомої раніше адреси. Деякі великі служби електронної пошти можуть використовувати велику кількість серверів з різними IP-адресами, і може виникнути така ситуація: кілька різних серверів намагаються надіслати одне і те ж повідомлення по черзі. Через принцип "сірого списку" це може спричинити великі затримки з доставкою листів.

Крім того, з розвитком анти-спам-програм, програми, що використовуються для масового розповсюдження спаму, постійно вдосконалюються. Легко реалізувати підтримку ретрансляції повідомлень, і це вдосконалення повністю виключає цей тип захисту.

2.3.4 Контроль масовості

Цей прийом передбачає виявлення великої кількості однакових або дещо різних повідомлень у поштовому потоці. Побудова ефективного "якісного" аналізатора вимагає величезного потоку пошти, тому ця технологія забезпечується великими компаніями, які працюють з великою кількістю електронної пошти, яку можна проаналізувати.

Перевага цієї технології: якщо система працює нормально, можна виявити велику кількість електронних листів. Недоліки: Перш за все, «великі» електронні листи можуть бути не спамом, а досить законними (наприклад, Ozon.ru, Subscribe.ru надсилає тисячі майже однакових електронних листів, але не спам) [42]. По-друге, спамери можуть використовувати розумні технології, щоб "прорвати" цей захист. Вони використовують програмне забезпечення, яке виробляє будь-який вміст-текст, графіку тощо. -У кожному спам-листі. Як результат, контроль якості не працює.

2.3.5 Перевірка заголовків листа

Спамери створили спеціальні програми для створення шкідливих повідомлень та їх негайного розповсюдження. Однак, оскільки спам не завжди відповідає вимогам поштового стандарту RFC, який описує формат заголовка, вони допускають помилки в дизайні заголовків. Для цих помилок ви можете порахувати спам.

Переваги цієї технології: процес виявлення та фільтрації спаму є прозорим, регулюється стандартами і досить надійним [43]. Недоліки цієї техніки: спамери

швидко вчаться, і в заголовках спаму стає все менше помилок. Тільки подальше використання цієї технології затримає вас не більше ніж на третину всього спаму.

2.3.6 Фільтрація за темою

Це також давня і зріла технологія. Перевірте специфічні для спаму слова, фрагменти тексту, зображення та інші особливості спаму в спамі. Фільтрація теми починається з аналізу теми електронного листа та тих частин, які містять текст (звичайний текст, HTML), але тепер фільтр спаму перевіряє всі деталі, включаючи графічні вкладення. В результаті аналізу можна створити текстовий підпис або розрахувати "вагу спаму" повідомлення [44].

Технічні переваги: гнучкість, можливість швидкого точного регулювання. Система, що працює з цією технологією, може легко адаптуватися до нових типів спаму, і вона рідко відрізняє спам від звичайної пошти. Недоліки цієї технології: зазвичай її потрібно оновити. Фільтри для спаму встановлюються спеціально навченим персоналом, іноді цілою лабораторією проти спаму. Така підтримка є дорогою, що впливає на вартість спам-фільтрів. Спамери винайшли деякі спеціальні прийоми, щоб обійти цю техніку: вони додають випадковий "шум" до спаму, що ускладнює пошук та оцінку характеристик спаму в повідомленні. Наприклад, у словах використовуються нелітерні слова (наприклад, при використанні цієї технології вигляд слова апельсин має вигляд: ap_el_sin на зображенні створюється змінний кольоровий фон тощо.

2.3.7 Тематична фільтрація: Баєсовський класифікатор

Статистичний алгоритм Байєса також призначений для аналізу вмісту і не вимагає постійної конфігурації. Для їх роботи потрібно зробити попередню підготовку. Після цього налаштуйте фільтр на тему літер, унікальну для конкретного користувача. Отже, якщо користувач працює та проходить навчання в системі освіти, його особисті повідомлення на цю тему не будуть визнані спамом

[46]. Для тих, кому не потрібно брати участь у тренінгу, статистичний фільтр перетворить такі повідомлення на спам.

Переваги цієї технології: персоналізація фільтрів для конкретних користувачів. Недоліки: байєсівські фільтри найкраще підходять для одного потоку пошти. Налаштувати Байєс з різнорідною поштою на корпоративному сервері великої компанії непросто. Але для малих та середніх компаній поєднання цієї технології з іншою технологією дозволить досягти найкращих результатів [41].

2.4 Опис предметної області

2.4.1 Метод дерев рішень

Дерево рішень відноситься до логічного методу класифікації. Дерево рішень - це ациклічний графік, згідно з яким об’єкти (у нашому випадку текстові документи), що описуються набором ознак, можна класифікувати [43]. Кожен вузол дерева містить умови для розгалуження на одному з елементів. Кожен вузол має стільки гілок, скільки значення вибраного елемента. У процесі оцінки безперервний перехід від одного вузла до іншого регулюється відповідно до значення характеристики об’єкта. Коли досягається один із листків дерева (кінцевий вузол), класифікація вважається повною [47]. Значення цього листа визначатиме клас, до якого належить об'єкт.

Насправді зазвичай використовується двійкове дерево рішень, в якому рішення про перехід краю приймається простою перевіркою наявності ознак у документі. Якщо значення ознаки менше певного значення, то вибирається одна гілка (якщо більша або дорівнює іншій). Коли мова йде про вибір найбільш підходящих атрибутів, зазвичай це стосується ключових слів. З цієї точки зору будь-яку частотну характеристику можна розглядати як змінну [48]. Потім вибір між двома найбільш підходящими ознаками полягає в оцінці ступеня кореляції між двома змінними. Тому, щоб вибрати відповідні ознаки на практиці, будь ласка, використовуйте різні критерії для перевірки гіпотези, тобто критерії, що

використовуються для кількісної оцінки ступеня кореляції між двома змінних, а 1 вказує на їх найбільшу залежність [47].

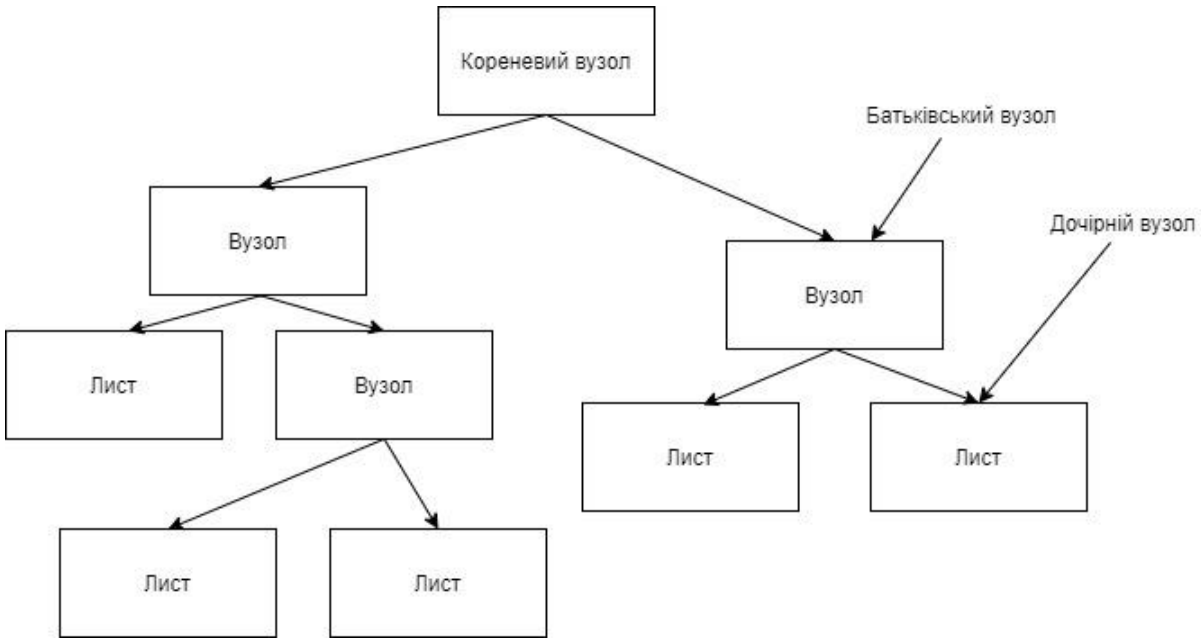

Рис. 2.2 – Приклад дерева рішень [45]

Для вивчення взаємозв'язку між двома змінними можна зручно використовувати табличну форму, щоб виразити загальний розподіл цих змінних. Це найбільш універсальний інструмент для вивчення статистичних взаємозв’язків, оскільки він може представляти змінні на будь-якому рівні вимірювання.

Таблиця взаємозв'язків зазвичай використовується для перевірки гіпотези про взаємозв'язок двох ознак з використанням різних статистичних стандартів: критерій Фішера (точний критерій Фішера), критерій консистенції Пірсона (критерій хі-квадрат), критерій Крамера, критерій Стьюдента (t-тест).). Але цей метод має великий недолік, тобто важко писати графіки для оцінки та нестабільності на великій кількості тексту.

2.4.2 Метод опорних векторів

Метод опорних векторних машин (SVM) є методом лінійної класифікації. В даний час цей метод вважається одним із найкращих методів [48]. Розглянемо безліч

документів, які потрібно класифікувати. Порівняємо це з набором точок розмірності | D |.

Якщо точки, що належать до різних категорій, можна розділити гіперплощиною у двовимірному випадку гіперплощиною є пряма лінія то зразки цих точок називаються лінійно неподільними [33].

Очевидним способом вирішення проблеми в цьому випадку є проведення лінії так, що всі значення одного класу розміщуються з одного боку, а всі позиції іншого класу - з іншого. Тоді, щоб класифікувати невідомі точки, досить знати, з якого боку лінії вони будуть входити [49].

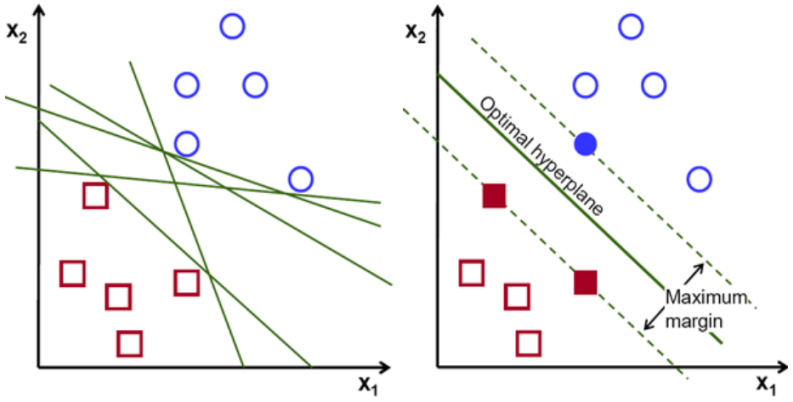

Рис 2.3 – Принцип побудови розділяючи гіперплощин [46]

Як правило, можна намалювати багато гіперплощин (ліній), які відповідають умовам. Очевидно, що лінію краще вибирати далі від доступної точки. У методі опорного вектора відстань між прямою і набором точок - це відстань між прямою і точкою, найближчою до точки. Це максимальна відстань у цьому методі. Гіперплощину, яка максимізує відстань від двох паралельних гіперплощин, називається роздільною площиною. Точка, найближча до паралельної гіперплощини, називається опорним вектором. Іншими словами, алгоритм працює за припущення, що чим більша різниця чи відстань між цими паралельними гіперплощинами, тим менша середня помилка класифікатора, оскільки максимізація відстані між класами допомагає класифікувати більш впевнено [50].

Насправді структура даних зазвичай невідома, і дуже рідко можна побудувати окрему гіперплощину, а це означає, що неможливо гарантувати лінійну роздільну здатність вибірки. Можуть бути файли, які присвоюють алгоритми одній категорії, але насправді вони повинні належати до протилежної категорії. Такі дані називаються викидами, і вони можуть спричинити методологічні помилки, тому їх краще ігнорувати. У цьому полягає суть проблеми лінійної нерозривності [34].

2.4.3 Метод k найближчих сусідів

Метод k-найближчого сусіда є одним з найпростіших алгоритмів класифікації і іноді використовується для задач регресії. Завдяки своїй простоті, це хороший приклад, з якого ви можете почати ознайомлення з областю машинного навчання, особливо із завданнями класифікації.

Щоб навчити класифікатора, ви повинні мати набір об’єктів, для яких класи визначені заздалегідь. Цей набір зразків називається навчальним зразком, і його маркування проводиться вручну за участю експертів у досліджуваній області. Наприклад, у завданні визначення зображень коментарів для заздалегідь зібраних коментарів людину запитують, чи є коментар зображенням одного з учасників обговорення, а саме завдання є прикладом двійкової класифікації. Класифікаційна задача багатокласова класифікація може мати більше двох класів, і кожен об’єкт може належати до кількох класів (середня класифікація) [51].

Щоб класифікувати кожен об'єкт тестового зразка, потрібно зробити наступне:

- Обчисліть відстань до кожного об’єкта в навчальній вибірці.
- Виберіть j об’єктів навчальної вибірки з найменшою відстанню. Клас об'єкта - це клас, який найчастіше з'являється серед j найближчих сусідів.

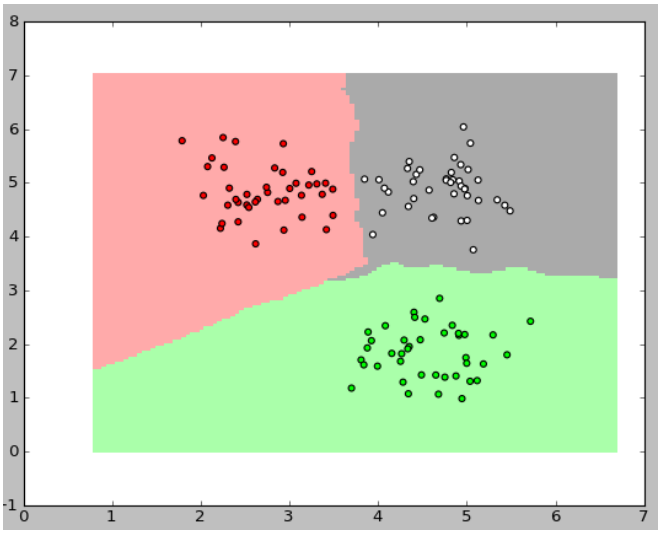

Рис. 2.4 – Принцип роботи методу [24]

kNN - це один з найпростіших алгоритмів класифікації, тому він, як правило, неефективний для практичних задач. На додаток до точності класифікації, проблемою цього класифікатора є швидкість класифікації: якщо в навчальній вибірці є N об'єктів, то в тесті відбирається M об'єктів, а розмірність простору -K, тест класифіковано Зразок операнда можна оцінити як O (K * M * N).

2.4.4 Баєсовський класифікатор

Цей метод простий (базовий алгоритм реалізації), зручний (дозволяє вам не потрібно використовувати "чорний список"), ефективний (до 95-97% спаму може бути перехоплений після навчання досить великої вибірки), і він може уникнути будь-яких помилки. Навч. Зазвичай на практиці всі ознаки вказують на його широке використання - він базується майже на всіх сучасних фільтрах спаму.

Однак у цього методу також є фундаментальний недолік: він базується на припущенні, що певні слова частіше зустрічаються у спамі, тоді як інші слова частіше зустрічаються в звичайних електронних листах. Якщо це припущення неправильне, воно є недійсним. Однак, як показала практика, навіть прочитавши та зрозумівши зміст такого спаму, людину не можна ідентифікувати «очима». Існує

метод байєсівського отруєння. Ідея цього методу полягає в тому, щоб додати багато зайвого тексту, іноді ретельно підібраного, щоб "обдурити" фільтр. Однак коли людина читає, такі листи будуть позбавлені сенсу, а отже, головна мета спамерів не буде досягнута [52].

Інший неосновний недолік, пов'язаний з реалізацією - цей метод застосовується лише до тексту. Знаючи це обмеження, спамери почали розміщувати рекламні повідомлення на малюнках. Текст у листі не існує або не має значення. На відміну від цього, ми повинні використовувати методи розпізнавання тексту («дорогі» програми, що використовуються лише тоді, коли це терміново потрібно), або використовувати старі методи фільтрації - «чорні списки» та регулярні вирази (оскільки такі букви зазвичай мають стереотипи)). Але в поєднанні з іншими методами його точність буде дуже високою [53].

Досліджуючи фільтр для кожного слова в тексті, розрахуйте та збережіть текст "ваги" для цього слова, щоб оцінити ймовірність спаму. У найпростішому випадку ця частота використовується як оцінка: "спам / весь вміст". У більш складних випадках текст можна попередньо обробити: відновити слова в первісному вигляді, видалити службові слова, розрахувати «вагу» всієї фрази, транслітерацію тощо.

При тестуванні нового тексту ймовірність "спаму" обчислюється за багатьма гіпотетичними формулами. У цьому випадку "гіпотеза" - це слово, і для кожного слова "надійність гіпотези" - це частка слова в тексті, а "залежність події від гіпотези" - це розрахована раніше "вага" цього слова ". Тобто в цьому випадку "вага" тексту - це середня "вага" всіх слів [53].

Присвоєння тексту "спаму" чи "не спаму" залежить від того, чи перевищує його "вага" стандарт, встановлений користувачем (зазвичай 60-80%). Після прийняття рішення щодо тексту "масштаб", що є частиною тексту, буде оновлений у базі даних [53].

Баєсові класифікатори відносяться до категорії машинного навчання. Найголовніше, що система, яка стоїть перед визначенням того, чи є лист спамом, попередньо навчена багатьма літерами. Серед цих листів точно відомо, яка буква є "спамом", а яка буква не є "спамом". Вже зрозуміло, що це відбувається під підготовкою вчителів, і ми тут як вчителі. Байєсівський класифікатор читає документи (у нашому випадку - літери) у вигляді набору слів, які, як кажуть, не залежать одне від одного (і тому мають однакову наївність).

Необхідно розрахувати рівень кожної категорії (спам / не-спам), а потім вибрати рівень, який отримує найвищий бал.

NBC базується на теоремі Байєса:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

- P(c|d) – ймовірність що документ d належить до класу c, саме її необхідно знайти.
- P(d|c) – ймовірність зустріти документ d серед всіх документів класу c.
- P(c) – безумовна ймовірність зустріти документ c в архіві документів.
- P(d) – безумовна ймовірність документа d в архіві документів.

Мета класифікації - зрозуміти, до якої категорії належить документ, тому нам потрібна не сама ймовірність, а найбільш вірогідна категорія. Байєсовський класифікатор використовує оцінку заднього максимального значення (максимального значення заднього оцінювання) для визначення найбільш вірогідної категорії.

$$c_{map} = \arg\max \frac{P(d|c)P(c)}{P(d)}$$

Грубо кажучи, це найбільш вірогідна категорія. Іншими словами, вам потрібно розрахувати ймовірності всіх категорій і вибрати категорію з найбільшою ймовірністю. Зверніть увагу, що знаменник (ймовірність документа) є константою і не впливатиме на рейтинг класу, тому ми можемо ігнорувати це у своїй задачі.

$$c_{map} = \arg\max [P(d|c)P(c)]$$

Алгоритм називається наївним класифікатором, оскільки, в більшості випадків, алгоритм може працювати за умови умовної незалежності, передбаченої базовою мовою програмування, тому ймовірність слова багато в чому залежить від контексту. Байєсівський класифікатор представляє документ як набір слів, ймовірності яких умовно не залежать один від одного. Цей метод іноді називають "моделлю мішка слів".

$$P(d|c) \approx P(w_1|c)P(w_2|c) \dots P(w_n|c) = \prod_{i=1}^{n} P(w_i|c)$$

Коли обсяг тексту великий - ймовірність мала, тому рекомендується використовувати логарифмічну властивість для перетворення формули log ab = log a + log b

Логарифм - монотонно зростаюча функція. Це видно з формули - ми шукаємо максимальне значення. Логарифм функції досягне свого максимального значення в тій самій точці (на абсцисі), що і сама функція. Оскільки змінюється лише значення, розрахунок спрощується. Основа логарифму не має значення, наприклад, може використовуватися десятковий або натуральний логарифм [54].

Підставивши отриманий вираз у формулу, описану нижче, а потім замініть його.

$$c_{map} = arg\max[\log P(c) + \sum_{i=1}^{n} \log P(w_i|c)]$$

Оцінка ймовірностей *P(c) і P(w_i|c)* виконується на вибірці. Імовірність класу записуються для оцінки як:

$$P(c) = \frac{D_c}{D}$$

де D_c це кількість записів, які належать до класу *c,* а D – вся кількість записів у вибірці. Оцінка імовірності слова в класі може відбуватись кількома засобами. Найпростіший це розрахувати взаємозв'язки

$$P(w_c|c) = \frac{W_{ic}}{\sum_{i\prime \in V} W_{i\prime c}}$$

Серед них W_ic - кількість випадків, коли слово з'являється у документі типу c. V-словник усіх унікальних слів.

Іншими словами, чисельник описує кількість випадків появи слова в документах класу (включаючи дублікати), а знаменник - це загальна кількість слів у всіх документах класу.

Під час процесу обчислення ми можемо зустріти слово, яке не знаходиться на стадії системи навчання. Це може спричинити нульовий бал, і документ не можна класифікувати як спам / не-спам. Скільки б ми не хотіли, ми не будемо викладати нашу систему на всіх можливих мовах. З цієї причини необхідно виконати згладжування та бути більш точним, внести невеликі виправлення до всіх ймовірностей слів у документі. Виберіть параметр $0 < \alpha \leq 1$ (якщо $\alpha = 1$, це згладжування Лапласа). Логічно кажучи, такий підхід змушує імовірнісні оцінки рухатись до малоймовірних результатів. Тому ймовірність слів, яких ми не бачили на етапі вивчення моделі, дуже мала, але все одно не нульова [55].

$$P(w_c|c) = \frac{W_{ic}+1}{\sum_{i\prime \in V}(W_{i\prime c}+1)} = \frac{W_{ic}+1}{|V|+\sum_{i\prime \in V} W_{i\prime c}}$$

Підставивши всі значення в формулу для оцінки можна отримати фінальну формулу по якій працює баєсівська класифікація

$$c_{map} = arg\max\left[\log\frac{D_c}{D} + \sum_{i=1}^{n}\log\frac{W_{ic}+1}{|V| + \sum_{i' \in V} W_{i'c}}\right]$$

Для реалізації байєсівського класифікатора необхідний навчальний зразок, у якому існує відповідність між текстовим документом та його категорією. Потім необхідно зібрати наступні статистичні дані із зразків, використаних на етапі класифікації [57]:

- Відносна частота категорій в тілі документа, тобто частота зустрічальності документів певної категорії;
- загальна кількість слів у документах кожної категорії;
- відносна частота слів у кожній категорії;
- Розмір зразка словника, тобто кількість унікальних слів у зразку.

Цей збір інформації буде називатися моделлю класифікатора. Потім на етапі класифікації необхідно розрахувати значення наступного виразу для кожної категорії та вибрати категорію з найбільшим значенням.

$$\log\frac{D_c}{D} + \sum_{i \in Q}\log\frac{W_{ic}+1}{|V| + L_c}$$

В цій формулі:

D_c – кількість записів в навчальній вибирці, які належать до класу *c*;

D – загальна кількість записів в навчальній вибірці;

$|V|$- кількість унікальних слів в усіх записах вибірки;

L_c – сумарна кількість слів в записах класу *c* в навчальній вибірці;

W_{ic} – скільки разів *i*-те слово зустрічалось в записів класу *c* в навчаючій вибірці;

Q – множина слів документа що класифікується.

Цей спосіб підходить для таких документів, як "мішок слів" або уніграма, тобто існує ймовірність кожного слова. Існує також спосіб розділити текст на два

слова "біграм" і три слова "триграма". Додавання такого розподілу покращить точність алгоритму, оскільки кожна фраза тоді має ймовірність бути "спамом" або "не спамом".

2.5 Опис існуючих алгоритмів

2.5.1 GPT-2

GPT був теоретично описаний Алеком Редфордом з OpenAI 11 червня 2018 року. Вже 14 лютого 2019 року була опублікована перша версія GPT-2 із найменшою кількістю параметрів (117 мільйонів), а 14 листопада 2019 року була опублікована повна версія (1,5 мільярда параметрів). У липні 2020 року стала доступна нова версія, GPT-3 із 175 мільярдами параметрів, але лише для закритого тестування. Станом на 23 вересня 2020 повний доступ до GPT-3 має лише компанія Microsoft. Отже, для цієї роботи використана GPT-2. [57]

GPT-2 побудований на архітектурі Transformers, проте використовує лише decoder-блоки.

GPT-2 працює не зі словами англійської мови, а з числовими кодуваннями слів, що впроваджують частину змісту слова. Розмірність цього кодування визначає розмірність і самої GPT-2. Так, наприклад, у найменшій моделі GPT-2 використовує 768 чисел для кодування одного слова, у найбільшій – 1600. Розмірність «словникового запасу» GPT-2 складає 50257 слів. До того, як почати тренування, тренувальний текст закодовується в числа, зрозумілі нейронній мережі; після генерації ці числа розкодовуються у зрозумілий читабельний текст.

У розділі 2.3 зазначалося, що однією з проблем визначення сенсу речення є контекстуальність змісту – себто в реченнях часто присутні слова, що або полісемічні («коса»), або не мають власного конкретного змісту та просто вказують на інше слово в реченні («воно»). На рисунку 2.5 показано, яким саме чином визначається контекст. Шар decoder-блоку Self-Attention оцінює, чи слово, що

сканується на поточний момент, вказує на якесь інше слово в реченні. Позначення Masked вказує на те, що сканування контексту йде лише з лівого боку – в будь-який окремий проміжок часу GPT-2 не знає «майбутнього» контексту слова.

Оцінка проходить наступним чином: робиться скалярний добуток вектору слова на вектор попередніх слів, після чого результати проходять через функцію softmax та отримуються імовірності зв'язку слова з іншими [58].

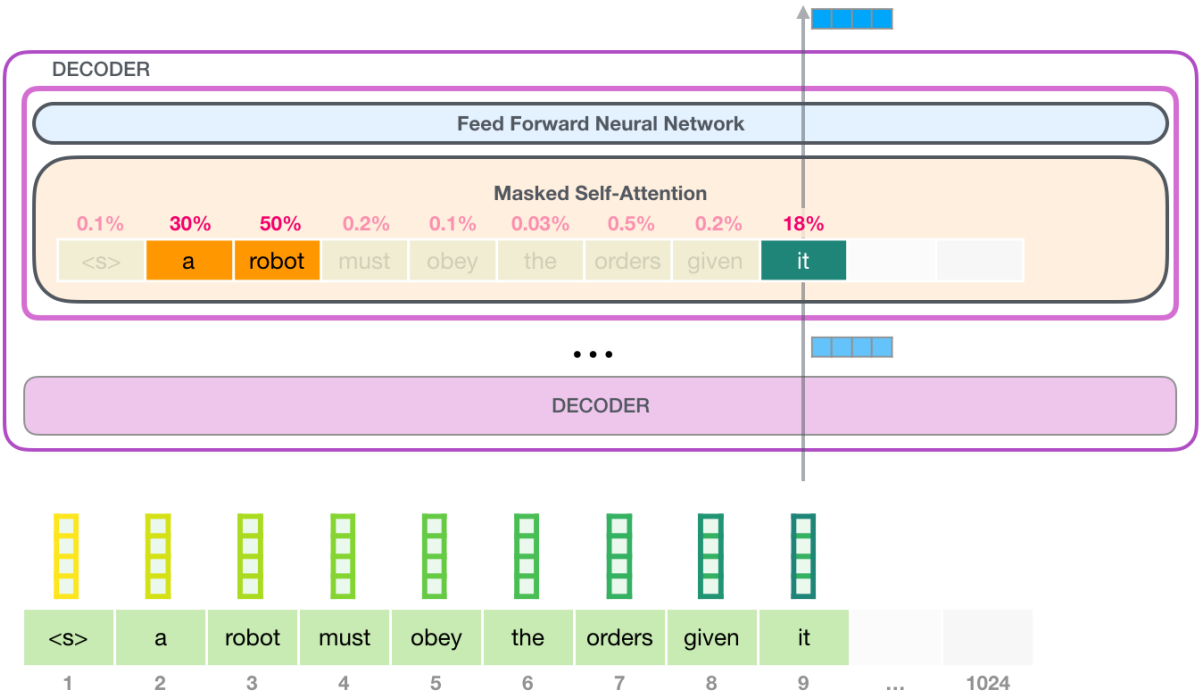

Рисунок 2.5 – Процес визначення мережею GPT-2 контексту слова в реченні

Як і традиційні мовні моделі (наприклад, модель Маркова), GPT-2 виводить лише одне слово за один раз. Під час генерації до ввідної послідовності додається останнє вивідне слово, після чого, опираючись на нову ввідну послідовність, GPT-2 видає наступне слово. GPT-2 не може дивитися на «майбутній» контекст слова, лише на минулий. Це повторюється задану кількість разів. Ця властивість називається авторегресивність [65].

Під час тренування в кінець кожного тексту додається токен <|endoftext|>, тому під час генерації його можна зазначити як стартовий токен.

Під час генерації GPT-2 пропонує декілька варіантів наступного слова, кожне з яких «підходить» відповідно до минулого контексту із певною ймовірністю. Яке

конкретно слово вибереться – це контролюється параметрами top_k або top_p. Параметр top_k відбирає певний відсоток найбільш імовірних слів. Зазначається, що «в цілому 40 є гарним числом» [58]. У випадку, якщо top_k дорівнює 40, GPT-2 відбирає 40 найімовірніших слів, що «найбільше» підходять за контекстом – а вже серед цих 40 слів наступне слово вибирається випадковим чином.

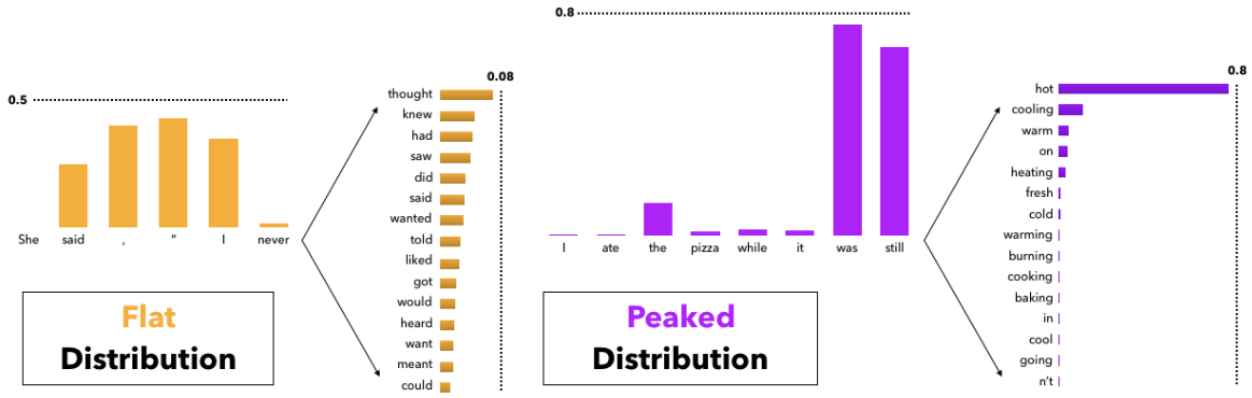

Рисунок 2.6 – Ілюстрація розподілення наступного слова. На малюнку жовте розподілення відповідає top_k, фіолетове – top_p

Параметр top_p відповідає імовірності, що слово «підходить» під контекст – при цьому слова із нижчою імовірністю відкидаються. Наприклад, якщо top_p дорівнює 0,9, то до уваги беруться лише ті слова, імовірність придатності яких перевищує 90%, а решта слів відкидається. [59].

Таким чином, параметр top_k бере до уваги певну кількість найімовірніших слів, а параметр top_p бере до уваги слова з певною мінімальною імовірністю.

2.5.2 BERT

BERT був теоретично описаний Якобом Девліном, Мінь Вей Чань та їхніми колегами з Google AI у статті від 11 жовтня 2018 року, а вже 2 листопада 2018 року BERT був викланий Google у відкритий доступ у двох варіантах: BERT Base із 110 мільйонами параметрів та BERT Large із 340 мільйонами параметрів. Обидва

варіанти були натреновані на двох текстових корпусах: BookCorpus, що складається з 11038 книг та 800 мільйонів слів, та корпусі текстів з англомовної Вікіпедії – 2,5 мільярдів слів [59].

Головною відмінною рисою BERT є двонаправленість його навчання. Зазвичай нейронні мережі тренуються на тексті у напрямку зліва-направо. Інколи комбінують результати навчання зліва-направо та справа-наліво. Тим часом, BERT визначає контекст слова на основі його оточення як зліва, так і справа по тексту.

Він побудований на архітектурі Transformers, проте включає в себе лише encoder-блоки.

BERT має дві моделі навчання: навчання з маскуванням (masked learning model, MLM) та передбачення наступного речення (next sentence prediction, NSP). Перша використовується для передбачення слова у реченні, а друга використовується для того, щоб дізнатися, чи друге речення є продовженням першого (складає цілісну сенсову одиницю), чи воно не має стосунку до першого.

MLM працює наступним чином. Під час тренування 15% слів замінюється на токен [MASK]. Модель намагається передбачити замасковане слово, спираючись на контекст до та після слова. З технічної точки зору це відбувається наступним чином (рис. 2.7):

1. Додається класифікаційний шар після шару «кодувальника» (encoder).
2. Перемножуються вихідні вектори на матрицю вкладання, перетворюючи їх у вимір словникового запасу.
3. Прораховування ймовірності кожного слова зі словникового запасу.

При цьому BERT відкидає значення передбачення для незамаскованих слів [67].

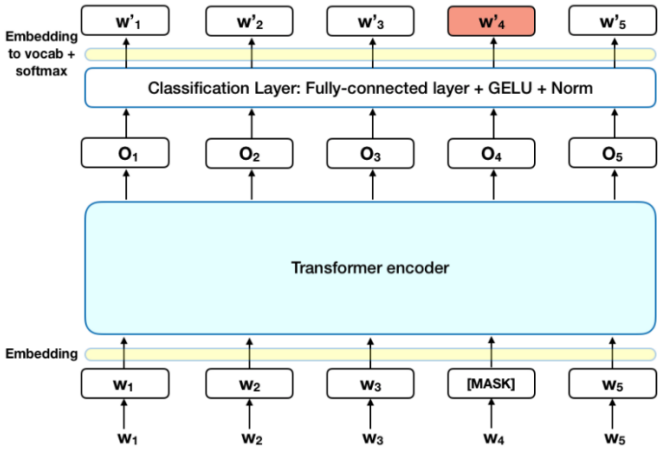

Рисунок 2.7 – Як працює MLM у BERT. Позначення: w_i – вхідні слова, O_i – вивід encoder-блоку, w'_i – кінцевий вивід.

NSP проходить наступним чином. Під час тренування 50% пар речень належать до одного тексту, а інші 50% пар речень взяті з абсолютно різних текстів та не мають жодного стосунку одне до одного. Для полегшення процесу навчання з послідовностями речень проводяться наступні маніпуляції (рис. 2.8) [71]:

1. Додається токен [CLS] на початок першого речення та токени [SEP] у кінець першого речення.
2. Додаються вкладання, що позначають речення А та речення Б.
3. До кожного слова додається вкладання, що позначає його позицію у послідовності речень.

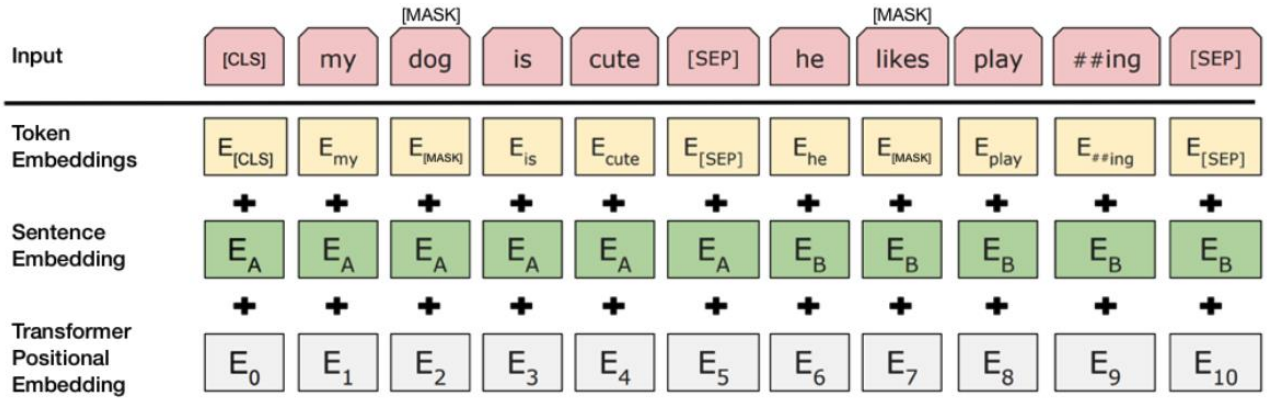

Рисунок 2.8 – Підготування речень до NSP-навчання BERT

Належність другого речення цілісному тексту визначається наступним чином [76]:

1. Уся ввідна послідовність проходить через модель Transformers.
2. Вивід токену [CLS] перетворюється на вектор розмірності 2x1, використовуючи вже натренований класифікаційний шар (з обрахованими вагами та зміщеннями).
3. Обрахування вірогідності належності до одного тексту функцією softmax [19].

Практичну користь для поставленої задачі несе саме перша модель навчання.

2.5.3 XLNT

XLNt був опублікований у статті 19 червня 2019 року, автором якої є Чжилін Янь, Зіхань Дай та інші з університету Карнегі-Меллон, сумісно з Куок Ле з Google AI Brain Team. Тоді ж було викладено і код мовою Python із моделлю XLNet-Large. Цікаво, що модель із більшою розмірністю – 340 мільйонів параметрів – було викладено раніше за модель XLNet-Base розмірністю в 110 мільйонів параметрів.

XLNet – це авторегресивна мовна модель, яка видає спільну ймовірність послідовності лексем, заснована на архітектурі Transformers-XL із рекурентністю. Його навчальна мета – обчислити ймовірність токену слова, обумовленого всіма перестановками токенів слів у реченні, на відміну від імовірностей на основі слів, що знаходяться ліворуч або лише тих, які розташовані праворуч від цільового токена. Більш докладно про це буде пояснено нижче.

Transformers-XL є покращенням архітектури Transformers – мережі, побудовані за цією архітектурою, можуть поєднувати інформацію з різних послідовностей (sequences) токенів. Також XLNet врахував інший недолік BERT – BERT передбачав токени незалежно один від одного.

Розглянемо речення «The boat was on the riverside». Як влучно ілюструє стаття [61], у реченні «The [mask] was [mask] on the riverside» BERT може вибрати як

варіанти, що складають одне смислове ціле: «The boat was beached on the riverside» та «The parade was seen on the riverside», так і помилковий варіант «The parade was beached on the riverside», тому що не враховує, як передбачені ними варіанти вкупі одне з одним впливають на контекст речення.

XLNet вирішує цю проблему таким чином, що вона вчиться не лише на певних цілісних реченнях, а й на «реченнях» зі словами, перемішаними у довільному порядку (рис. 2.9).

Таким чином, із тренувальним часом ця нейронна мережа вчиться розпізнавати контекст слів безвідносно до їхнього порядку в реченні. Такий підхід, за словами авторів, дозволяє вичавити більше інформації з одного й того самого тренувального корпусу.

Автори виклали [22] цю ідею в формулі (2.2):

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\mathbb{E}_{z \sim Z} \left[\sum_{t=1}^{T} \log \left[Pr\left(x_{z[t]} | x_{z[<t]} \right) \right] \right] \right] \tag{2.2}$$

У цій формулі критерій $\hat{\theta}$ знаходить такі параметри моделі θ, щоб максимізувати імовірність токенів $x_{z[t]}$ у послідовності довжини T, враховуючи попередні токени $x_{z[<t]}$, де $z[t]$ – це t-ий елемент перестановки $\mathbf{z}$ індексів токенів, а $z[< t]$ – попередні елементи перестановки. Сума логарифмів імовірностей означає, що для будь-якої однієї перестановки модель є належним чином авторегресивною, оскільки є добутком ймовірності для кожного елемента в послідовності. Математичне сподівання щодо всіх перестановок у Z показує, що модель навчена бути однаково здатною обчислювати ймовірності для будь-якого токена з урахуванням будь-якого контексту. [61]

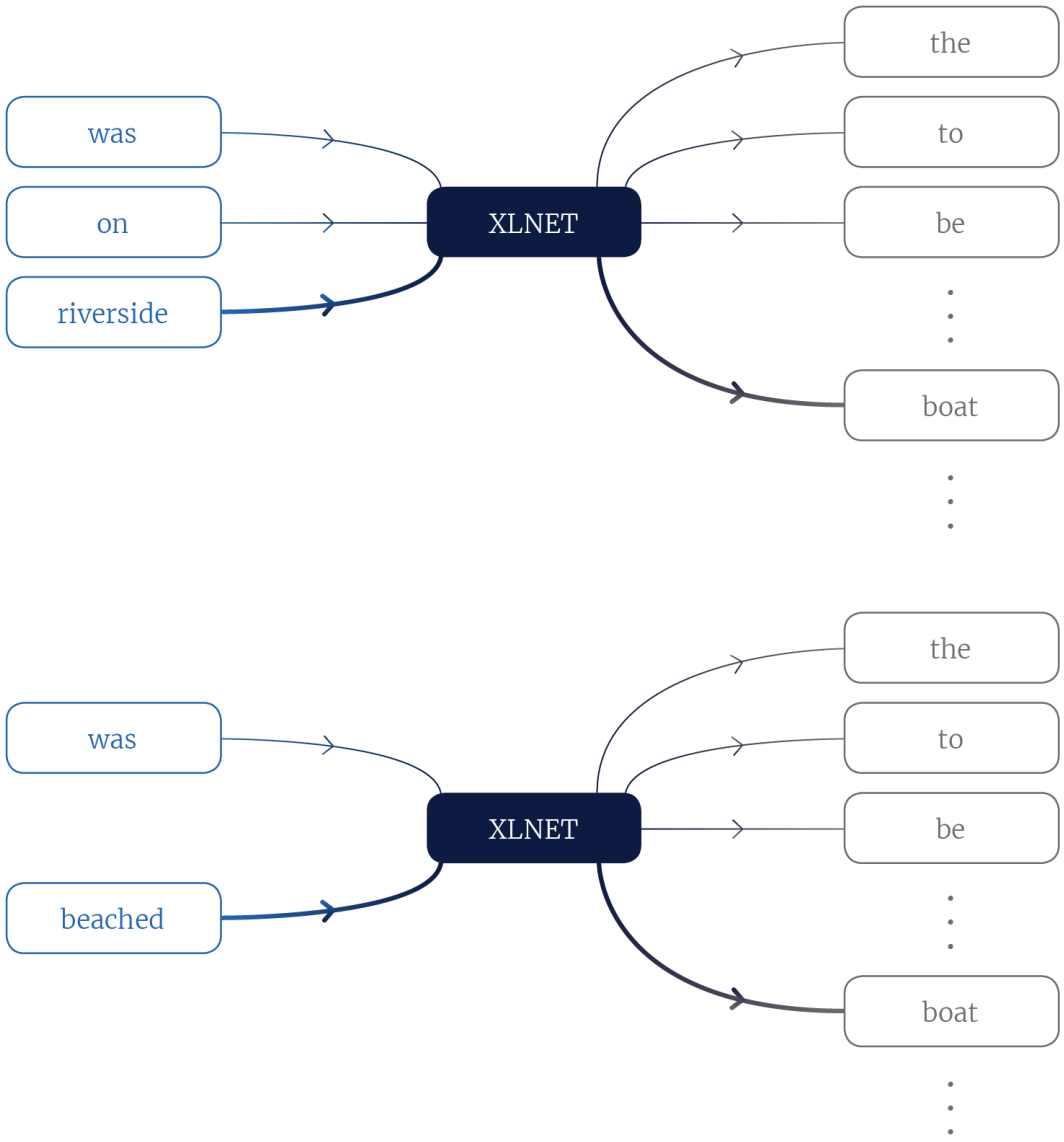

Рисунок 2.9 – XLNet вчиться на текстах із реченнями зі словами, перемішаними у довільному порядку

Тим не менш, у цій формулі неясно, яким чином модель знає про відносне положення слів у реченні. Якщо не надавати цієї інформації, то модель думатиме, що всі подані слова мають однакову ймовірність появи одне після одного. Натомість, автори прагнули створити модель, яка б могла передбачити імовірність появи слова із певним поданим порядковим номером у реченні.

Рішенням цієї проблеми є двопотоковий механізм самоуваги. Кожна позиція токену i має два пов'язаних вектори на кожному шарі самоуваги m: $\mathbf{h}_i^m$ та $\mathbf{g}_i^m$. Вектори $\mathbf{h}$ належать потоку вмісту (Content Stream), тоді як $\mathbf{g}$ вектори належать до

потоку запитів (Query Stream). Вектори потоку вмісту ініціалізуються словесними токенами, до яких додані токени з позначеннями позицій. Вектори потоку запитів ініціалізуються загальним вектором вбудовування **w**, доданим до позиційних вбудовувань. Зверніть увагу, що **w** однаковий, незалежно від токена, і тому не може використовуватися для розрізнення токенів [62].

На кожному шарі кожен вектор вмісту, $\mathbf{h}_i$, оновлюється з використанням тих **h**, які залишаються немаскованими самі по собі. Оновлення використовує вектори вмісту як запит, ключ і значення.

На відміну від цього, на кожному шарі кожен вектор запиту $\mathbf{g}_i$ оновлюється за допомогою незамаскованих векторів вмісту та самого себе. Оновлення використовує $\mathbf{g}_i$ як запит, тоді як $\mathbf{h}_j$ використовуються як ключі та значення, де j – індекс немаскованого токену в контексті i.

Розглянемо речення «This is a sentence». На рисунку 2.10 показано, як розраховується запит g_4^m для 4-го токена на m-му шарі самоуваги. Це показує, що g_4^m – це сукупність (is+2), (a+3) та позиції 4, що і є контекстом, необхідним для обчислення ймовірності токену «sentence».

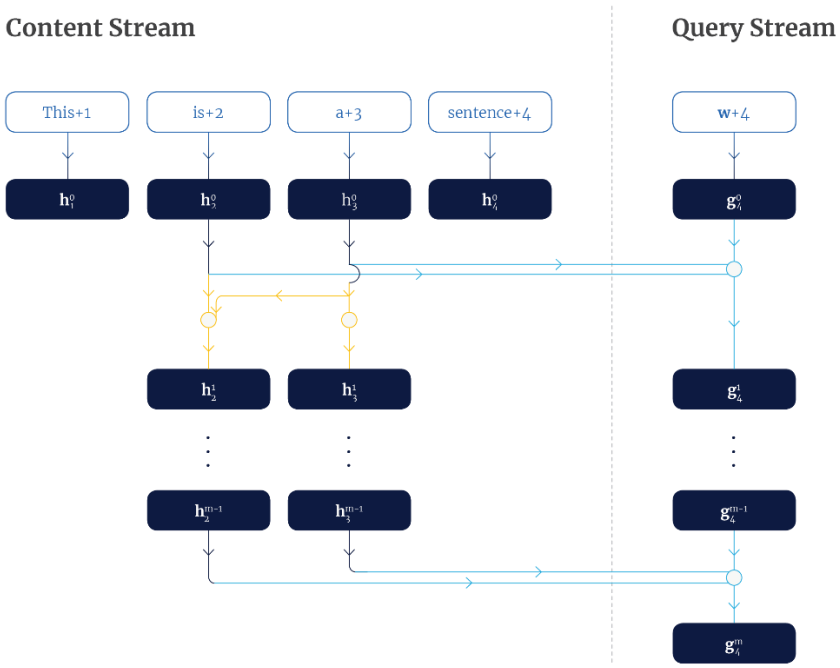

Рисунок 2.10 – Обрахування мережею XLNet імовірності появи четвертого токену в реченні «This is a sentence».

Стрілки вказують на потік інформації від векторів. Там, де лінії перетинаються по колу, виконуються та агрегуються операції запиту / ключа / значення до уваги. Жовті рядки представляють оновлення у потоці вмісту для третього символу (залежить лише від нього самого) та другого символу (залежить від нього самого та третього символу). Світло-сині рядки представляють оновлення в потоці запитів (залежить від нього самого і другого та третього символів із потоку вмісту).

Ця властивість XLNet дозволить побудувати на її основі двонаправлений генератор тексту природною мовою [62].

2.6 Стандарти системи запобігання вторгнень

2.6.1 Стандарт ISO 27039

Описуються основні принципи розгортання системи виявлення та запобігання вторгнень. А саме її вибір, розгортання, експлуатація, який зображений на рисунку. 2.11. У той же час надаються довідкову інформацію, для отримання рекомендації.

Перед вибором системи виявлення та запобігання вторгнень організації повинні знати не тільки те, коли трапився інцидент, а також, яким чином, це трапилося, які вразливості були використані, і які заходи захисту були реалізовані. А також, які заходи захисту будуть реалізовані в майбутньому для зниження ризиків.

Організації мають виявляти та запобігати мережевим вторгненням. Для того щоб отримати максимальний захист від продукту IDS/IPS, її вибір, розгортання та експлуатація повинна бути здійснена кваліфікованими та досвідченими спеціалістами.

Тільки в такому випадку буде можливим отримання інформації від систем виявлення та запобігання вторгнень і використання цієї інформації для забезпечення безпеки всієї інформації та комунікаційної інфраструктури.

Стандарт ISO 27039 призначений в першу чергу, щоб допомогти:

a) Організаціям задовольнити наступні вимоги:

- організація повинна впровадити процедури та інші елементи управління для швидкого виявлення та реагування на інциденти безпеки;
- організація здійснює процес моніторингу й огляду і контроль подій безпеки для виявлення інцидентів безпеки.

b) Організаціям забезпечити контроль безпеки для вирішення наступних завдань:

- виявлення несанкціонованих дій з обробкою інформації;

система повинна відстежувати і записувати події інформаційної безпеки.

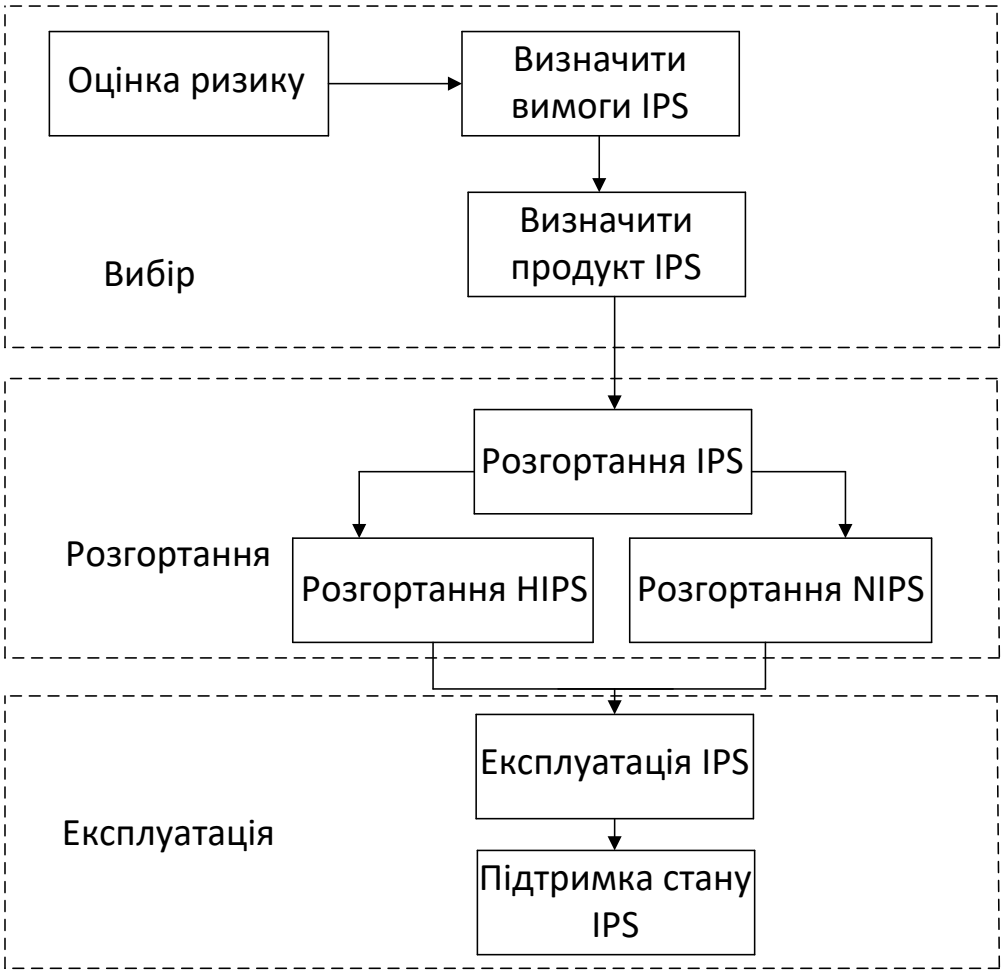

Рисунок 2.11 – Вибір, розгортання, експлуатація IPS

- Оператор повинен використовувати журнал подій, щоб переконатися, що подія відбулася саме на визначеному вузлу у локальній мережі.

- організація повинна відповідати всім відповідним правовим вимогам, що застосовуються для контролю і реєстрації діяльності;
- система моніторингу повинна бути використана для перевірки достовірності контролю, здійснюваного для перевірки моделі політики доступу відповідності.

Організаціям повинно бути до відома, що виконання вищевказаних вимог не є єдиним або абсолютним рішенням.

Крім того, цей стандарт не призначений, щоб бути частиною критеріїв оцінки відповідності, таких як системи управління інформаційною безпекою (СУІБ) сертифікації [63].

2.6.2 Стандарт PCI DSS

Стандарт безпеки даних індустрії платіжних карток був розроблений для заохочення та поліпшення безпеки даних власників карток та спрощення прийняття узгоджених заходів безпеки даних у всьому світі [54].

PCI DSS забезпечує базові технічні та експлуатаційні вимоги для захисту даних облікових записів власників карток. PCI DSS застосовується всіма організаціями, які беруть участь в обробці платіжних карт, включаючи продавців, емітентів та постачальників послуг. PCI DSS також застосовується до всіх інших об'єктів, які зберігають, обробляють або передають дані власника картки та/або чутливі дані аутентифікації.

Стандарт має наступні вимоги з використанням системи запобігання вторгнень [58]:

1) Переглядати наступні параметри, принаймні щодня:

- усі події безпеки;
- журнали всіх компонентів системи, які зберігають, оброблюють, передають дані власника карти та/або чутливі дані аутентифікації;
- журнали всіх критично важливих компонентів системи;

- журнали всіх серверів і систем компоненти, які виконують функції безпеки наприклад, брандмауери, системи виявлення вторгнень, системи запобігання вторгнень.

2) Використовувати системи виявлення/запобігання вторгнень для виявлення та запобігання загрозам в мережі. Проводити моніторинг мережі, яка стосується периметру середовища даних власників карток, а також критичні точки середовища даних власників карток. Попереджати персонал про можливу компрометацію в мережі або в критичних точках. Слідкувати за тим, щоб сигнатури систем виявлення/запобігання вторгнень були в оновленому стані.

3) Переконатися, що системні конфігурації та мережеві діаграми відповідають тому, що методи захисту (такі, як системи виявлення вторгнень, системи запобігання вторгнень) здійснюють контроль та моніторинг мережевого трафіку:

- по периметру середовища даних власників банківських карток;
- у критичних точках середовища даних власників карток.

4) Переконатися у коректності системних конфігурацій у IDS/IPS, провести інструктаж для персоналу у разі виявлення інциденту безпеки зазначеними системами.

5) Переконатися, що конфігурація, підтримка та оновлення систем виявлення/запобігання вторгнень здійснюються згідно з документацією постачальника зазначених систем для забезпечення оптимального захисту.

6) Увімкнути сповіщення про інциденти безпеки на системах моніторингу мережі, а також в системі виявлення/запобігання вторгнень.

7) Переконатися шляхом спостереження та огляду, що система виявлення/запобігання вторгнень обробляє, реагує та надсилає сповіщення про інциденти безпеки [64].

2.6.3 Методологія OWASP

В даний час існує велика кількість організацій, що займаються безпекою у веб середовищі. Однією з найбільш відомих організацій є Open Web Application Security Project (OWASP) - онлайн співтовариство, яке займається створенням вільно розповсюджуваних статей, методологій, інструментів і технологій в області безпеки веб-додатків [56].

В рамках спільноти OWASP ведеться робота більш ніж над 140 проектами, кожен з яких визначається набором пов'язаних завдань, певним планом розвитку і командою розробників. Всі проекти спрямовані на створення інструментів і документації з інформаційної безпеки в наступних категоріях [58]:

- захист від атак і вразливостей у веб додатках;
- виявлення атак і недоліків в уже існуючих системах;
- проектування і реалізація програмного забезпечення з урахуванням вимог інформаційної безпеки.

Так, як організація спрямована на захист веб-додатків, вона також створює різноманітні продукти для захисту веб-додатків. Один з таких є OWASP AppSensor. Ця програмна утиліта вміщує в собі функціонал системи виявлення/запобігання вторгнень та активно застосовуються для проведення тестування веб-додатків [65].

Висновки до розділу 2

У розділі 2 були проаналізовані теоретичні відомості про механізми ідентифікації шкідливих листів, алгоритми, а також відповідні міжнародні стандарти, щодо вибору, експлуатації та супроводу цих систем. Однією із найголовніших компонентів архітектури захищеної мережі.

Були опрацьовані ймовірні варіанти реалізації та визначені можливі обмеження для них, що можуть сприяти виникненню небажаних подій. В результаті два з трьох розглянутих методів мали серйозні недоліки, а саме нестійкість методу на великий вибірці даних а також ускладнення їх розробки з ростом кількості листів.

З інтенсивним ростом розвитку антиспам систем, все ж залишаються певні проблеми такі, як генерування великої кількості хибних подій, збільшення

пропускної здатності, покращення відмовостійкості системи. Один з варіантів їх вирішення є застосування ефективного алгоритму та ефективний з точки зору виявлення ознак шкідливого трафіку.

РОЗДІЛ 3
ВДОСКОНАЛЕННЯ СИСТЕМИ ІДЕНТИФІКАЦІЇ ШКІДЛИВИХ ЛИСТІВ ЗА РАХУНОК ВИЗНАЧЕННЯ ЕФЕКТИВНОГО АЛГОРИТМУ РОЗПІЗНАВАННЯ

3.1 Опис алгоритмів виявлення ознак мережевих атак та їх проблематика

1. Проблематика ефективності у алгоритмах виявлення ознак мережевих атак.

З огляду на останні досягнення інформаційних та комунікаційних технологій, а також всебічно зростаючу потребу в мережевих технологій, інформаційна безпека стає дуже гострою проблемою. Одночасно з цим спостерігається широке застосування Інтернету організаціями та підприємствами різної форми власності для обміну даними різної форми конфіденційності. Тому надзвичайно важливим є захист інформаційних ресурсів, що є актуальною проблемою сьогодення.

В останні кілька років, атаки у вигляді шкідливого мережевого трафіку та мережевих атак, який зловмисники використовують для компрометації інформації на комп'ютерні мережі постійно зростають. На теперішній час існують різні інструменти та механізми, що спрямовані на забезпечення інформаційної безпеки. Одна з таких рішень є система виявлення та запобігання мережевих вторгнень.

2. Опис алгоритмів.

Традиційні системи для виявлення шкідливого трафіку з метою запобігання вторгнень в основному використовують "сигнатурний" метод, який вимагає створення унікальної риси (ознаки) для кожного типу атаки на інформаційну систему. При цьому кожна нова сигнатура спочатку додається до бази даних IPS, а потім виконується порівняння ознак вхідного трафіку з відповідними еталонними ознаками, які закладені в банк, для подальшого розпізнавання та його виявлення у мережі.

Створення сигнатур зазвичай здійснюється за допомогою аналізу відповідних експертів та має постійно оновлюватися. При цьому існує декілька проблем, пов'язаних з цим методом:

- система повинна бути спочатку скомпрометована для того, щоб були відомі основні ознаки шкідливого трафіку;
- для кожної нової кібератаки потрібно визначення нової сигнатури.

Більш того, в окремих сценаріях кібератак, IPS, яка базується на сигнатурному методі не гарантує достатньо швидкого виявлення ознак шкідливого трафіку, що пов'язане з витраченим часом на розпізнавання однієї ознаки і як наслідок, деякі пакети шкідливого трафіку можуть бути пропущені, що приведе до прихованої компрометації інформаційної системи. Як і кожен програмний продукт, IPS потребує значних обчислювальних ресурсів, а саме великих об’ємів оперативної пам'яті та потужності процесору.

На сьогодні вже існує низка алгоритмів ідентифікації мережевих атак (IMA) за якими працюють IPS. Алгоритми розпізнавання мережевих атак є одними з найважливіших компонентів для ефективної роботи IPS. В цьому розділі представлені сучасні алгоритми для апаратних пристроїв системи запобігання вторгнень [66].

3. Алгоритм паралельних обчислень Блума.

Цей алгоритм використовує фільтр Блума для окремих шаблонів ознак, які мають різну довжину. Загалом, фільтр застосовує декілька методів гешування, що призводить до ефективного використання простору збереження геш-значень шаблону ознак. Метод за яким працює алгоритм здатний обчислювати 4 байта за одиницю часу з пропускною здатність 2,47 Гбіт/с. Факт того, що для кожної іншої довжини шаблону вимагається окремий фільтр Блума, робить алгоритм неефективним для використання для шаблонів довжина, яких може досягати тисячі байтів. Фільтр Блума використовує техніку випадковості для перевірки потоків даних на наявність певних ознак. З одного рядка мережевих даних X, фільтр обчислює k геш-функцію та надає результат гешу, який варіюється від 1 до m. Потім

встановлюються *k* біти у вектор *m* довжини, які відповідають адресам значенням гешу *k*. Така процедура називається програмуванням фільтра та проводиться для кожного рядка мережевих даних. Процедура запиту є подібною до процедури програмування, в ній рядок мережевих даних, вводиться до фільтру. Фільтр Блума генерує *k* значення гешу за допомогою геш-функції аналогічно до процесу програмування. Потім розглядаються біти в *m*-бітовому векторі в місцях, відповідних *k* геш-значень. Якщо принаймні один з *k* бітів не встановлений до відповідності певного шаблону, тоді рядок мережевих даних зазначається, як нешкідливий та вибуває з подальшого аналізу. У випадку коли всі *k* біти встановлені до відповідності шаблону ознак, тоді рядок мережевих даних оголошується належним до шаблону із певною імовірністю. Ця невизначеність у наборі виходить з того, що *k* біти у m-бітовому векторі можуть бути встановлені для будь якого з рядків *n*. Виявлення не встановленого біта, означає, що рядок мережевих даних не належить до шаблону ознак. Це пояснює наявність хибних спрацьовувань та відсутності хибно негативних спрацьовувань (false negatives) [67]. Відсоток хибних спрацьовувань визначається формулою 3.1:

$$f = (1 - e^{-nk/m})^k, \tag{3.1}$$

де *n* кількість рядків запрограмовані фільтром Блума;

k – кількість бітів у рядку;

m – бітовий вектор;

Значення *f* можна зменшити вибравши найоптимальніші значення *m* та *k* для заданого розміру набору елементів *n*. Зрозуміло, що значення *m* повинні бути достатньо великими в порівнянні з розміром рядка мережевих даних. Також для даного відношення *m/n* ймовірність хибних спрацьовувань може бути зменшена шляхом збільшення числа геш-функцій *k*.

В оптимальному випадку, коли ймовірність хибних спрацьовувань мінімізована по відношенню з *k*. Отримуємо наступну функцію мінімізації хибних спрацьовувань у формулі 3.2 [85]:

$$f = (m/n)\ln 2, \tag{3.2}$$

де *n* кількість рядків запрограмовані фільтром Блума;

m – бітовий вектор;

Це відповідає ймовірності хибного спрацьовування, які можемо спостерігати за формулою 3.3.

$$f = (1/2)^k, \tag{3.3}$$

де k – кількість бітів у рядку [86];

Відношення m/n інтерпретується, як середня кількість бітів, споживаних одним шаблоном ознак. Варто відзначити, що ця вимога незалежна від фактичного розміру шаблону. В оптимальному варіанті ймовірність хибного спрацьовування експоненційно зменшується з лінійним зростанням відношення *m/n*. Це також значить, що число геш-функцій, відповідно числа випадкових пошуків у бітовому векторі необхідних для запиту одного шаблону ознак, пропорційне *m/n*.

Заданий набір ознак згрупований за їх довжиною, зберігається в наборі апаратних засобів паралельних фільтрів Блума. Кожен із цих фільтрів містить шаблони ознак певної довжини. Фільтри використовуються для моніторингу мережевого трафіку та роботи з рядками мережевих даних відповідної довжини, які показані на рисунку 3.1.

Кожний рядок мережевого трафіку перевіряється у відповідності з фільтрами Блума. Якщо знайдений рядок співпадає із значенням одного із фільтрів, то він оголошується, як ймовірна загроза. Потім такі рядки досліджуються в аналізаторі, який визначає, чи є даний мережевий трафік шкідливим чи хибним спрацьовуванням.

Коли досліджуваний рядок визначений, як шкідливий, для мережевого пакету, який містить цей рядок, виконується одна з дій функціоналу IPS: видалення, блокування, повідомлення адміністратору [68].

Переваги:

- відсутність негативно хибних спрацьовувань;
- робота на швидкості до 2 Гбіт/с.

Недоліки:

- кожний шаблон сигнатури вимагає окремий фільтр Блума;
- неможливість розпізнавання шкідливого трафіку у якого довжина певних ознак сягає більше тисячі байтів;
- велика кількість хибних спрацьовувань.

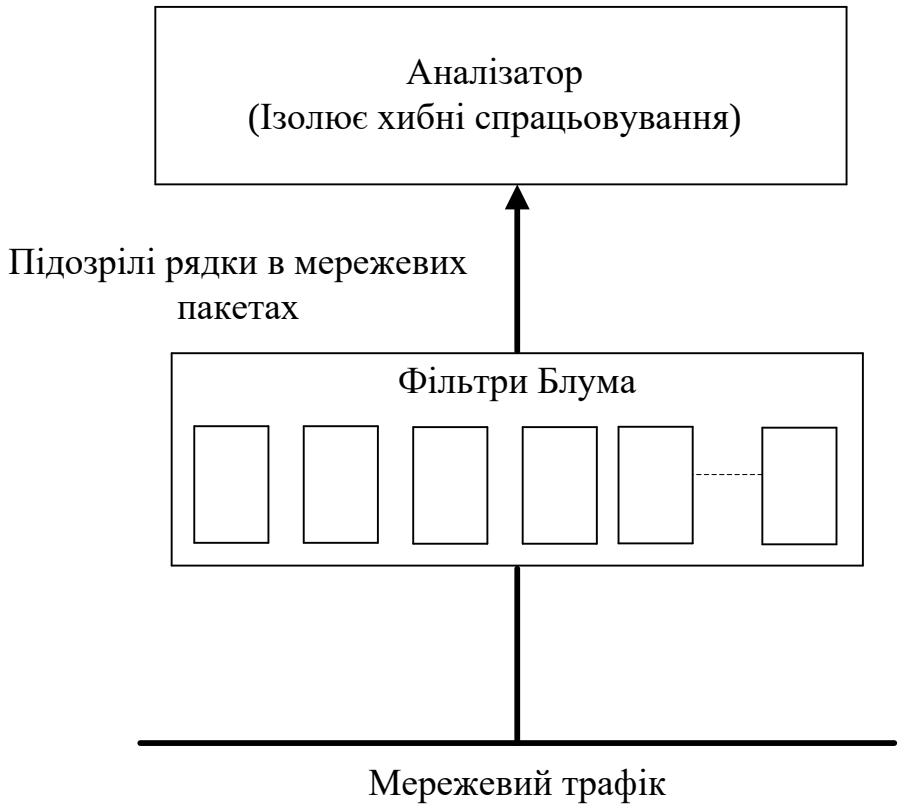

Рисунок 3.1 – Фільтри Блума сканують мережевий трафік на швидкості гігабітної мережі за визначеними ознаками

Алгоритм порівняння мережевого процесору (МП)

АМП - це алгоритм на основі зсуву, який використовує мережевий процесор з механізмом гешування за допомогою пам'яті. Алгоритм використовує вікно зсуву *w,* який рухається від крайнього лівого байту до крайнього правого байту рядка даних. Алгоритм порівняння шаблонів мережевого процесору підтримує тільки прості шаблони ознак шкідливого трафіку не корелюючи їх. Таблиця зсуву\Skip Distance Table (STD), яка реалізована в цьому алгоритмі має розмір ((2^8)^w).

Система виявлення та запобігання вторгнень Snort була одна з перших, де реалізовувався алгоритм АМП.

Реалізація алгоритму АМП є досить простою. АМП ґрунтується на наступних простих міркуваннях: Для довільного шаблону ознак шкідливого трафіку $Pj = a_0^j$, $a_1^j, \ldots, m - 1^j$. Якщо послідовні байти вікна зсуву w для вхідних даних *T* розташовується в s, де $t_{s+I} \ldots t_{s+w}-1 != a^j{}_0 \ldots a^j{}_{w-1-i}$, $i = 0,1,2, \ldots, w-1$. Тоді *w* не містить перших та послідовних байтів *Pj*, де $1 \leq i \leq w$. З цього слідує що, послідовні байти *w* не містять будь-якого попереднього шаблону ознак шкідливого трафіку P_j, тоді послідовні байти можуть бути пропущені під час пошуку. У випадку, якщо послідовні байти w для вхідних даних *T* знаходяться в s, які відповідають шаблону ознак P_j, то порівняння решти (*m*-*w*) байтів є правильним вішенням.

В процесі порівняння використовується вікно зсуву з довжиною *w,* яке зсувається від крайнього лівого байту до крайнього правого байту вхідних даних *T*. Кожного разу, коли вікно зміщується, здійснюється спроба визначити *S*, *w* послідовний байт, який міститься у вікні, вміщує $a^j{}_0 \ldots a^j{}_{k-1}$ шаблону ознак $P_{j,}$, де $1 \leq k \leq w$. Якщо такого шаблону не існує, де перші *k* послідовні байти містяться в *S*, тоді вікно може бути зсунуто в бік вправо на *w* байт. Проте, якщо шаблон P_j існує, де $S_{w-k} \ldots S_{w-1} = a_{j0} \ldots a^j{}_{k-1}$, тоді вікно буде переміщено вправо на *w*-*k* байтів. Навіть якщо знайдено послідовні байти в *T*, які будуть відповідати шаблону P_j, помилкові спрацювання будуть існувати. Цей алгоритм характеризується тим, що метод виявлення ознак за допомогою вікна зсуву буде зменшувати навантаження на систему. Якщо *w* є достатньо малої довжини (наприклад 3) STD, може перерахувати всі можливі варіанти з певної кількості послідовних байтів. Для прикладу позначимо

w = 3. Такий метод виконується інтуїтивно, але часом він є ефективним. Наприклад якщо шаблон ознаки шкідливого трафіку системи запобігання вторгнень містить ‘abcd’ ідентичний до {0x61, 0x62, 0x63, 0x64} в кодуванні ASCII, тоді таблиця запису буде, як 0x000061 до 0xFFFF061. Відповідно, коли *S* потрапить у цей проміжок, він буде зміщений на 3 байти, так, що нове розташування першого байту буде відповідати ‘a’ [47]. Робота алгоритму показана на рисунку 3.2.

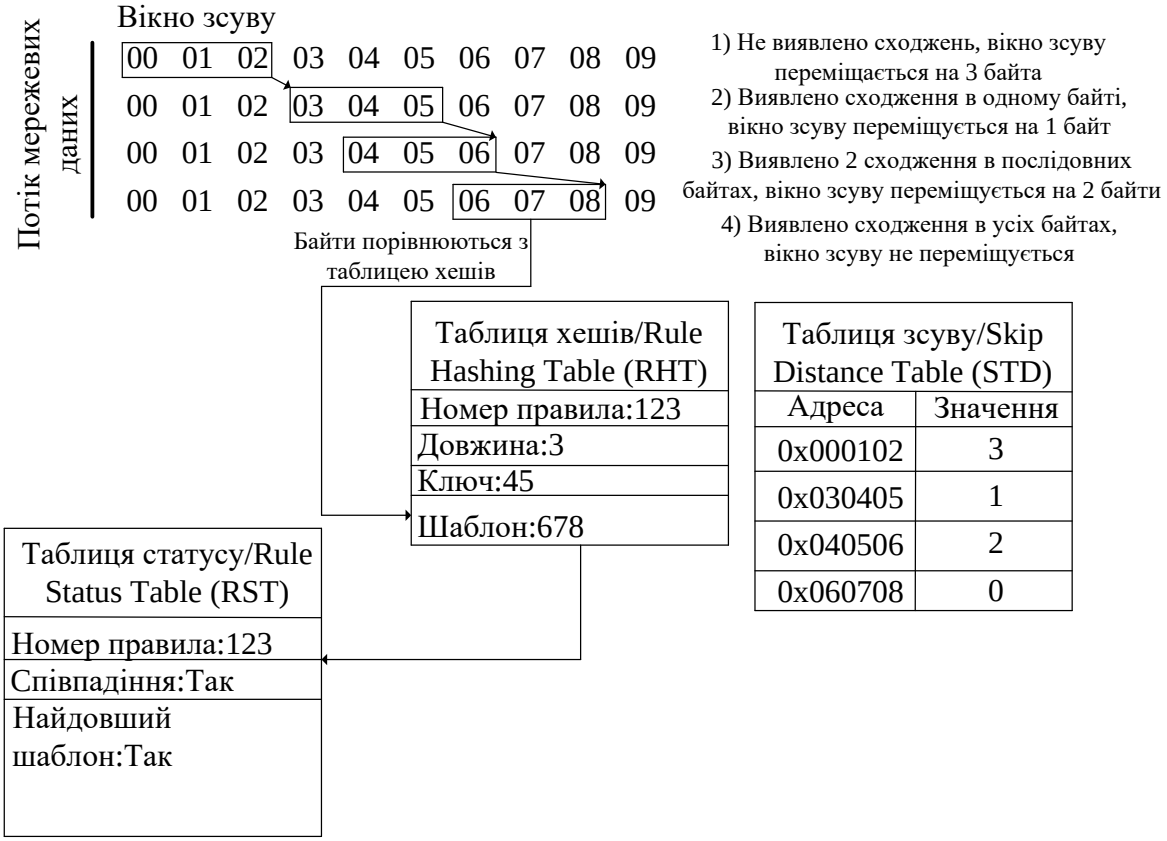

Рисунок - 3.2 Приклад роботи алгоритму

Згодом записи таблиці адреси 0x616263 встановлюються в нуль. Тоді *S* ідентичний 0x616263 відповідає першим трьом послідовним байтам рядка, який містить "abcd". Інші записи таблиці встановлюються рівними 3. Якщо *S* потрапляє в цю категорію, можна перемістити вікно зсуву на три байти вправо. На (рис 3.2) показано, як вікно зсуву *w* рухається зі значеннями входу в SDT.

Алгоритм поділяється на попередню обробку та обробку на час виконання. Впродовж попередню обробку конструюються необхідні таблиці правил і таблиці пошуку при одночасному процесі обробки процесів, які можуть бути завантажені та ідентифіковані.

STD таблиця використовується для пошуку числа байтів на які вікно зсуву має перемітитись, а також таблиця перевіряє чи S відповідає шаблону ознак. Розмір SDT 2^{w*8}.

Таблиця гешу\Rule Hash Table (RHT) зберігає рядки сигнатур\ознак шкідливого трафіку. RHT містить посилання на рядки колій, ідентифікатори для гешу певної ознаки, вміст рядка, ідентифікатор правила.

Таблиця статусу правила\Rule Status Table (RST) призначена для кількох цілей [75]:

- для запису того, чи шаблон був ідентифікований раніше;
- для розміщення шаблонів декількох змістів;
- для збереження пріоритетів правил.

RST має такий самий розмір, як і кількість елементів рядка вмісту в RHT, і порядок записів RST точно такий же, як і ідентифікатор правила в RHT.

Під час процесу попередньої обробки створюються таблиці STD, RHT та RST. У таблицях STD ініціалізується значення при якому буде здійснюватися порівняння. У таблицях RHT записуються правила, для кожного гешу даних. У RST таблицях встановлюється прапор MATCH при якому перевіряється чи була ідентифікація певного шаблону ознак раніше. Це виконується для того щоб швидше розпізнавати ознаки шкідливого трафіку.

Процес обробки алгоритму досить простий. Спочатку вікно зсуву вирівнюється з першим байтом вхідного корисного навантаження. Потім вікно зсуву переміщується на стільки байтів, скільки написано у таблиці SDT. Якщо збігу не було знайдено вікно зсуву знову переміщується на кількість байтів згідно з SDT. Якщо збіг був знайдений у одному із байтів тоді зсув збільшується на один і знову перевіряється вміст. Якщо співпадіння були виявлені, зсув збільшується на один.

Якщо співпадіння не було виявлено, то байти, які співпадають надсилаються у таблицю RST і перевіряється чи був такий шаблон шкідливого трафіку виявлений. Якщо шаблону не було виявлено раніше, то відбувається точна відповідність між ознаками шкідливого трафіку та решти вмісту трафіка. Якщо шаблон був виявлений раніше, тоді пакет позначається, як шкідливий і система запобігання вторгнень виконує наступну із процедур (сповіщення адміністратора, блокування, видалення) [69].

Переваги:

- Є оптимальним алгоритмом для простих IDS\IPS (наприклад Snort).

Недоліки:

- застосування маленького вікна зсуву призводить до генерації великої кількості хибних спрацьовувань;
- не справляється з великою кількістю трафіку.

4. Алгоритм відповідності шаблонів TCAM

TCAM - це розширений чіп пам'яті, який може зберігати три значення для кожного біта: нуль, один і "не визначений". Сучасні алгоритми зіставлення зразків, що використовують TCAM, були представлені Лакшманом. Запропонований алгоритм розміщує набір сигнатур атак у TCAM і розгортає простий алгоритм узгодження шаблонів. Ключ w байтів послідовно будується з пакету (шляхом перенесення тексту на один байт за один раз), і TCAM шукає відповідність. Алгоритм може працювати на швидкості 2 Гбіт/с.

Ternary Content Addressable Memory (TCAM) - це тип пам'яті, який може виконувати паралельний пошук на високих швидкостях. TCAM складається з набору записів. Верхній запис TCAM має найменший індекс, а нижній - найбільший. Кожен запис являє собою вектор бітів комірок, де кожна комірка може зберігати один біт.

Таким чином, запис TCAM може використовуватися для зберігання рядка. TCAM працює наступним чином [78]: з урахуванням вхідного рядка він паралельно порівнює цей рядок з усіма записами в його пам'яті і повідомляє про один запис,

який відповідає введеному. Час пошуку є детермінованим для будь-якого введення. На відміну від двійкової CAM, яка має тільки два стани: 0 або 1, кожна комірка в TCAM може приймати одне з трьох станів: 0, 1 або ?. Зі станом "не визначений". При такому варіанті алгоритм TCAM виконує додаткові перевірки. Також через стан "не визначений" один вхід даних може співпадати з кількома записами TCAM.

Припустимо, що ширина TCAM становить w байтів. Розглянемо спочатку простий випадок, коли всі шаблони є простими детермінованими, довжина яких коротше або дорівнює w байтам. Таке рішення реалізується в TCAM, кожен з яких займає один запис. Якщо малюнок коротше, ніж w байтів, то розміщується його незаповнені біти на «?». Шаблони повинні бути організовані відповідно до їх довжини в порядку спадання. Це пояснюється тим, що TCAM лише повідомляє про перший результат відповідності, якщо є кілька збігів, все одно позначається перший. Наприклад, якщо шаблон “ABC” введений в нижній індекс TCAM, узгодження “ABC” включає узгодження короткого шаблону “AB”. Якщо ми розміщуємо шаблони в іншому порядку, ми не можемо зробити висновок про відповідність більш довгого шаблону з відповідності більш короткому шаблону. Таким чином, ми можемо пропустити деякі відповідні результати.

Процес пошуку шаблонів у пакеті виглядає наступним чином: Перші w байти в пакеті відображаються в TCAM. Якщо виявлена відповідність до шаблону ознак шкідливого трафіку, алгоритм повідомляє про збіг, як зображено на рисунку 3.3. Цей процес повторюється, поки не буде перевірений весь пакет.

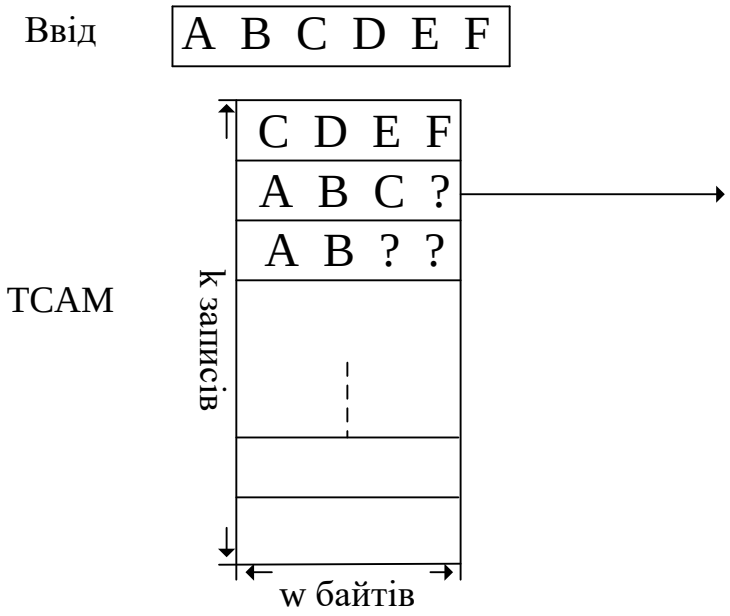

Рисунок 3.3 – Робота алгоритму TCAM

Простий шаблон P з m байтів може бути записаний як $P = b_1 b_2 \ldots b_m$, де кожне b_i являє собою байт. Довжина шаблону m може бути різною для кожної ознаки. b_i може бути двох форм: детермінованої форми або недетермінованої форми.

Для детермінованої форми кожен біт b_i є 0 або 1. Це одне конкретне значення $2^8 = 256$ можливих значень. Наприклад, $b_i = 0100\ 0001$ буква (а) [70].

Для недетермінованої форми b_i може бути будь-яким значенням. Нижче наведено два види груп байтів:

Байти нечутливого алфавіту: $b_i = \{a, A\}, \ldots., b_i = \{z, Z\}$.

Вільні байти (*): b_i може бути будь-яким з 2^8 значенням.

Переваги:

- Працює на швидкості до 2 Гбіт\с.

Недоліки:

- непрозоре налаштування шаблонів ознак шкідливого трафіку;
- можливі пропуски шкідливих ознак;
- не працює з великими об’ємами трафіку.

5. Scilit-learn

Автором Девідом Корнапеуом була розроблена програмна бібліотека в 2008 році. На сьогоднішній день задіяно задіяно більше 29 учасників і інвестиції від компаній, як INRIA, Google, Tinyclues і Python Software Foundation.

Дана бібліотека реалізаціє певний список функцій\алгоритмів для машинного навчання через інтерфейс мови програмування Python. Описана бібліотека розповсюджується з-під ліцензії "License BSD " охоплюючи комерційне і академічне використання scikit-learn.

Scikit-learn створена з допомогою SciPy, яка повинна бути встановлена перед застосуванням scikit. Описаний стек включає в себе [71]:

- SciPy: бібліотека наукових застосунків для верхньорівневої мови програмування Python;
- Matplotlib: бібліотека на основі мови Python. Створена для візуалізації даних графіки;
- NumPy: модуль мови Python, який додатково підтримує великі масиви і матриці.
- IPython: фреймворк для мови Python, який надає огляд, додаткові символи, підказки і автоматичне доповнення;
- Sympy: модуль для обчислень;
- Pandas: обчислення і аналіз структури даних.

```
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
# fit a CART model to the data
model = DecisionTreeClassifier()
model.fit(X, y)
print(model)
# make predictions
expected = y
predicted = model.predict(X)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

Рис 3.4 – Код методу Decision Tree [65]

```
from sklearn import metrics
from sklearn.svm import SVC
 # fit a SVM model to the data
model = SVC()
model.fit(X, y)
print(model)
# make predictions
expected = y
predicted = model.predict(X)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

Рис 3.5 – Код методу Support Vector Machine [66]

```
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
# fit a k-nearest neighbor model to the data
model = KNeighborsClassifier()
model.fit(X, y)
print(model)
# make predictions
expected = y
predicted = model.predict(X)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

Рис. 3.6 – Код методу kNN [67]

Головною метою бібліотеки scikit-learn є бібліотека з високим рівнем стабільності і підтримки, який потрібний для підприємницьких систем, що означає, велику увагу для питань зручності застосування коду, документації та швидкодії роботи даної бібліотеки.

Бібліотека scikit-learn першочергово орієнтована на моделювання даних, взаємодію і колекціонування даних. Для такої мети доцільніше застосовувати бібліотеки NumPy і Pandas. Ось кілька популярних функціональних областей, в яких scikit-learn допомагає вирішувати поставлені завдання [72]:

- Формалізація: для зведення в категорії даних, наприклад, метод k-середніх.
- Перехресна перевірка: для надання оцінки продуктивності роботи моделі на довільних даних.
- Комплекси даних: для тестових шаблонів даних і для створення груп даних з визначеними властивостями для проведення дослідження поведінки моделі.

- Зменшення розмірності: для скорочення кількості метрик для графіку та відбору атрибутів, таких як, функція основних компонентів.
- Множинне навчання: для нелінійного скорочення розмірності даних.

6.NLTK.

Бібліотека NLTK є провідною платформою для створення програм Python для роботи з даними мови людей. Бібліотека надає прості у використанні інтерфейси для більш ніж 50 лексичних ресурсів, а також комплекс бібліотек для обробки тексту, класифікації, позначення, розбору та семантичних міркувань, обгортки для промислових бібліотек.

```
import nltk
def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    sentence = """Arthur didn't feel very good."""
    tokens = nltk.word_tokenize(sentence)
    text = "|".join(str(x) for x in tokens)
    return bytes(text.encode('utf8'))
```

Рис 3.7 – Python код для NTLK [68]

NLTK підходить для лінгвістів, інженерів, студентів, педагогів, дослідників і користувачів галузі. Бібліотека доступна для низки ОС таких як: Windows, Mac OS X і Linux. Найкраще, NLTK - це вільний проект, орієнтований на громаду.

NLTK називався «чудовим інструментом для навчання та роботи в комп'ютерній лінгвістиці з використанням Python» і «дивовижною бібліотекою для гри з природньою мовою» [73].

Обробка природних мов з Python забезпечує практичне введення в програмування для обробки мови. Написана творцями NLTK, вона веде читача через основи написання програм Python, роботи з корпусами, категоризації тексту, аналізу мовної структури та багато іншого. Онлайн-версія книги була оновлена для Python 3 і NLTK 3

7. TextBlob

TextBlob - це бібліотека Python (2 і 3) для обробки текстових даних. Вона надає простий API для занурення в загальні завдання з обробки природних мов

(NLP), такі як часткове мовлення, екстракція іменника, аналіз настроїв, класифікація, переклад і багато іншого.

Особливості [74]:

- Вилучення іменника
- Класифікація (алгоритм байєса)
- Позначення частин мовлення
- Переклад та виявлення мов
- Токенізація (розділення тексту на слова та пропозиції)
- Слова і фрази
- Слово перегину (плюралізація і сингулярность) і лемматизація
- Коригування орфографії
- Інтеграція WordN*et*

8. Polyglot

Бібліотека Polyglot також має корисний функціонал і може бути додана до модулю в майбутньому, для реалізації підтримки багатьох мов черз наступні особливості [75]:

- Токенізація (165 мов).
- Виявлення мови (196 мов).
- Розпізнавання названої сутності (40 мов).
- Частина міток мовлення (16 мов).
- Аналіз настроїв (136 мов).
- Вбудовування слів (137 мов).
- Морфологічний аналіз (135 мов).
- Транслітерація (69 мов).

9. CoreNLP

Бібліотека CoreNLP пропонує набір інструментів для опрацювання мови людей. Надає базові форми слів, частини мови, чи є вони назвами компаній, людей, нормалізує дати, позначає структуру речень у термінах фраз і синтаксичних

залежностей, часи і числові величини, вказати які іменні фрази відносяться до одних і тих самих сутностей, вказують на настрої, витягують окремі або відкриті зв’язки між об'єктними сутностями, отримують цитату людей і т.д.

CoreNLP буде гарним вибором для розробки якщо потрібно [76]:

- Інструментарій NLP з широким спектром інструментів аналізу.
- Швидкий, надійний аналізатор для довільних текстів, широко використовуваний у виробництві.
- Сучасний, регулярно оновлюваний пакет із загальною якісною аналітикою тексту.
- Підтримка кількох основних (людських) мов.
- Доступні API для більшості сучасних мов програмування.
- Можливість працювати як простий веб-сервіс.

Мета бібліотеки CoreNLP полягає в тому, щоб зробити легким, для застосування набору інструментів лінгвістичного аналізу до частини тексту. Інструментальний конвеєр можна запускати на фрагменті звичайного тексту лише двома рядками коду. CoreNLP розроблений так, щоб бути дуже гнучким і розширюваним. За допомогою однієї опції можна змінити, які інструменти слід увімкнути та вимкнути. Стенфордський CoreNLP об'єднує багато інструментів, включаючи мовлення (POS), ідентифікатор імені об'єкта (NER), синтаксичний аналізатор, систему дозволу коренерії, аналіз настроїв, навчання початкових шаблонів і інструменти відкритої видобування інформації. . Крім того, конвектор анотатор може включати додаткові користувацькі або сторонні анотатори [77]. Аналізи CoreNLP забезпечують основні будівельні блоки для додатків, які розуміють більш розвинутий рівень і специфіку домену.

10. Gmail API

Компанія Google вирішила відкрити для сторонніх додатків доступ до вмісту поштових скриньок Gmail і представила програмні інтерфейси Gmail API.

Gmail API надають доступ до окремих ресурсів, таким як History, Messages, Labels, Drafts, Threads або History. Наприклад, сторонній додаток може відправити

запит на доступ тільки на відправку листів (але не читання), або тільки на читання (не відправляючи), або тільки на зміну міток для листів і ланцюжків, або тільки на пошук конкретних листів і ланцюжків [78].

Gmail - найпопулярніших поштовий сервіс в світі. Точна кількість користувачів приблизно невідома, але в 2015 році Google оцінювала аудиторію в 625 млн чоловік, при цьому зазначила, що Gmail є поштовим провайдером для 71 з 100 найбільших університетів в світі і 5 млн комерційних компаній.

Це означає, що Gmail зараз – скоріш за все, друга платформа за кількістю активних користувачів після Facebook, що має однозначно залучити розробників додатків. Звичайно, створювати додатки на базі поштового сервісу здається дивним, але Google хоче поекспериментувати.

Користь від сканування поштової скриньки можуть отримати найрізноманітніші програми. Наприклад, будь-яка програма може автоматично погоджувати email, не змушуючи людину натискати по посиланню підтвердження. Достатньо всього лише сканувати папку «Вхідні» на появу відповідного тексту. Або, наприклад, програма обліку витрат може автоматично імпортувати з папки «Вхідні» чеки із зазначенням статей витрат і витраченої суми в інтернет-магазинах. Або програма може від імені користувача висилати листи, відповідно до заданих умов [79].

Крім того, за допомогою Gmail API можна створювати різні розширення для стандартного веб-інтерфейсу Gmail. Наприклад, кнопку для «заморозки» папки «Вхідні» на 20-50 хвилин або тиждень, коли людина не хоче відволікатися. В даному сценарію можливо придумати масу ідей. У тому числі робити розширення, які просувають власний бренд. Скажімо, розширення, яке здійснює запрограмовані дії при одержанні листа з заздалегідь встановленим змістом.

Хоча перед розробниками відкриваються нові можливості, але користувачам доведеться уважніше ставитися до видачі дозволів на доступ сторонніх додатків до персональної інформації. Втім, в цьому відношенні інтерфейси Gmail API краще в безпеці, ніж стандартний IMAP, тому що можуть обмежувати повноваження

сторонніх програм необхідним мінімумом. Також ця бібліотека була обрана через те що Gmail є найпопулярнішим поштовим додатком в світі, тому всі потенційні користувачі будуть орієнтовані на пошук модулів які будуть підтримувати саме Gmail.

```
def main():
    creds = None
    if os.path.exists('token.pickle'):
        with open('token.pickle', 'rb') as token:
            creds = pickle.load(token)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'credentials.json', SCOPES)
            creds = flow.run_local_server()
        with open('token.pickle', 'wb') as token:
            pickle.dump(creds, token)
    service = build('gmail', 'v1', credentials=creds)
```

Рис 3.8 – Python код для підключення Gmail API [69]

3.2 Опис алгоритмів, які можуть бути застосовані в системи запобігання вторгнень

Алгоритми, які описані вище, використовувались або використовуються в системах IPS. Проте вони не можуть гарантувати достатній рівень захисту від мережевих атак, оскільки кожен з алгоритмів має свої недоліки. Тому одним з необхідних рішень є пошук нових алгоритмів РМА, які можуть бути застосовані в системах запобігання вторгнень [80].

3.2.1 Алгоритм Байєса

Формула Байеса - одна з основних теорем елементарної теорії ймовірностей, яка дозволяє визначити ймовірність якої-небудь події за умови, що сталося інша статистично взаємозалежна з ним подія. Іншими словами, за формулою Байеса можна більш точно перерахувати ймовірність, взявши до уваги як раніше відому інформацію, так і дані нових спостережень. Формула Байєса може бути виведена з

основних аксіом теорії ймовірностей, зокрема з умовної ймовірності. Особливість теореми Байеса полягає в тому, що для її практичного застосування потрібна велика кількість розрахунків, обчислень.

При виникненні теореми Байеса ймовірності, що використовуються в теоремі, піддавалися цілої низки імовірнісних інтерпретацій. В одній з таких інтерпретацій говорилося, що виведення формули безпосередньо пов'язаний із застосуванням особливого підходу до статистичного аналізу. Якщо використовувати Байєсова інтерпретацію ймовірності, то теорема показує, як особистий рівень довіри може кардинально змінитися внаслідок кількості наступили подій. У цьому полягають висновки Байеса, які стали основоположними для байєсівської статистики. Однак теорема використовується не тільки в Байєсова аналізі, а й активно застосовується для великого ряду інших розрахунків [81].

Психологічні експерименти показали, що люди часто невірно оцінюють ймовірність події, на основі отриманого досвіду (апостеріорна ймовірність), оскільки ігнорують саму ймовірність припущення (апріорна ймовірність). Тому правильний результат за формулою Байеса може сильно відрізнятися від інтуїтивно очікуваного.

Теорема Байєса названа в честь її автора Томаса Байеса - англійського математика і священика, який першим запропонував використання теореми для коригування переконань, ґрунтуючись на оновлених даних [81; 82]. Формула Байєса у матричному вигляді показана формулою 3.4.

$$K = \frac{e^{(S_{ei}^{*} P^{-1} S_{ei})}}{\sqrt{(2\pi)^{p} D}}, \qquad (3.4)$$

де S_{ei} – усереднений вектор ознак і-го елементу;

P^{-1} – обернена кореляційна матриця; e – експонента;

D – значення дисперсії випадкової величини;

p – кількість ознак.

Формула Байєса дозволяє "переставити причинність": розрахувати ймовірність того, що подія спричинена цією причиною, виходячи з відомих фактів події. Безумовна ймовірність справедливості гіпотези називається апріорі (наскільки великою є причина), тоді як враховується факт події та умовний задній стан (наскільки великою є причина, заснована на даних події).

3.2.2 Відстань Махаланобіса

Відстань Махаланобіса - міра відстані між векторами випадкових величин, що узагальнює поняття евклідового відстані [85]. Запропоновано індійським статистиком Махаланобіса в 1936 році. За допомогою відстані Махаланобіса можна визначати схожість невідомої і відомої вибірки. У формулі 3.5 відображена відстань Махаланобіса у матричному вигляді

$$K = \arg\min_i (S - S_{ei})^* P_i^{-1} (S - S_{ei}), \tag{3.5}$$

де P^{-1} – обернена кореляційна матриця;

S_{ei} - усереднений вектор ознак і-го еталону;

S – вектор ознак;

arg min – мінімальне значення імовірності.

Відстань Махаланобіса також можна визначити як міру розбіжності між двома випадковими векторами з одного розподілу ймовірностей з матрицею кореляції.

Якщо матриця кореляції є одиничною матрицею, то відстань Махаланобіса стає рівним відстані Евкліда. Якщо матриця кореляції діагональна, то міра відстані буде носити назву нормалізована відстань Евкліда:

Відстань Махаланобіса від багатовимірного вектору $x = (x_1, x_2, x_3 \ldots x_N)^T$ до множини із середнім значенням $\mu = (\mu_1, \mu_2, \mu_3 \ldots \mu_N)^T$ матрицею коваріації S визначається, як показано у формулі 3.6.

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1}(\mu)}, \tag{3.6}$$

де S^{-1} – обернена матриця коваріації;

x^T – багатовимірний вектор;

μ^T – множина.

Відстань Махаланобіса також можна визначити, як міру розбіжності між двома випадковими векторами x та y з одного розподілу ймовірностей з матрицею коваріації S.

Розрахунки приведені у формулі 3.7.

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}, \tag{3.7}$$

де, S^{-1} – обернена матриця коваріації;

x^T , y^T – випадкові вектори.

Для прикладу розглянемо задачу визначення ймовірності того, що деяка точка в N-вимірному евклідовому просторі належить множині, яка задана набором точок, належать даній множині. Знайдемо центр мас множини. Інтуїтивно зрозуміло, що чим ближче задана точка до центру мас, тим більша ймовірність того, що вона належить множині.

Однак також варто враховувати, на якого розміру області розосереджені точки безлічі, щоб зрозуміти, наскільки значимо відстань між заданою точкою і центром мас.

Найпростіший підхід полягає в обчисленні середньоквадратичного відхилення точок безлічі від центру мас. Якщо відстань між заданою точкою і центром мас менше середньоквадратичного відхилення, то можна зробити висновок, що ймовірність приналежності точки безлічі висока. Чим далі точка, тим більша ймовірність того, що вона не належить множині.

Цей інтуїтивний підхід можна визначити математично через відстань між заданою точкою і безліччю за формулою обчислення середньоквадратичного відхилення 3.8.

$$Q = \frac{x - \mu}{\sigma}, \tag{3.8}$$

де σ_i середньоквадратичне відхилення *x* від *y* у вибірці;

μ – випадкова множина.

За допомогою підстановки цього значення в нормальний закон розподілу можна знайти ймовірність приналежності точки безлічі.

Недолік такого підходу полягає в використанні припущення про те, що точки безлічі сферично розподілені навколо центру мас. Якщо ж розподіл явно не сферичний, то було б природним враховувати в ймовірності приналежності не тільки відстань до центру мас, а й напрямок на нього. У напрямку короткої осі еліпсоїда задана точка повинна бути ближче до центру мас, щоб належати безлічі, в той час як в напрямку довгої осі вона може бути далі.

Для запису цього в математичному вигляді еліпсоїд, найкращим чином представляє імовірнісний розподіл безлічі, може бути заданий матрицею кореляції множини. Відстань Махаланобіса - це просто відстань між заданою точкою і центром мас, поділене на ширину еліпсоїда в напрямку заданої точки [85].

Щоб використовувати відстань Махаланобіса в задачі визначення приналежності заданої точки одному з N класів, потрібно знайти матриці кореляції всіх класів. Як правило, це робиться на основі відомих вибірок з кожного класу. Потім необхідно підрахувати відстань Махаланобіса від заданої точки до кожного класу і вибрати клас, для якого це відстань мінімально. Використовуючи вірогідну інтерпретацію, можна показати, що це еквівалентно вибору класу за допомогою методу максимальної правдоподібності [83; 84].

3.2.3 Алгоритм Кейпона

Алгоритм Кейпон або метод максимальної правдоподібності був запропонований в 1969 році. Використовуючи аналогію частотного і просторового спектрів, метод був запропонований також для оцінювання координат сигналів [54; 56; 57]. Формула обчислення алгоритму Кейпона приведена у 3.9.

$$K=\arg\max_i \frac{1}{S^*_{ei}(P^{-1})S_{ei}}, \tag{3.9}$$

де P^{-1} – обернена кореляційна матриця;

S_{ei} - усереднений вектор ознак і-го эталона;

arg min – мінімальне значення імовірності.

3.2.4 Алгоритм теплового шуму

Тепловий шум - це збалансований шум, спричинений тепловим рухом носіїв заряду в провіднику, який спричинить різницю потенціалів на кінці провідника.

Тепловий шум буде виникати в будь-якому провіднику струму з активним опором і пов’язаний з хаотичним рухом рухомих носіїв заряду, тому на кінцях провідника з’являться коливання напруги. Реактивний опір - ємність та індуктивність - не можуть бути джерелом теплових шумів.

У металах завдяки високій концентрації провідних електронів і короткій довжині вільного пробігу теплова швидкість електронів у багато разів перевищує швидкість спрямованого руху в електричному полі. Отже, потужність теплового шуму не залежить від прикладеної напруги, струму чи частоти [85; 86]. Формула алгоритму теплових шумів наведена в 3.10.

$$K = \arg\max_i \frac{1}{S^*_{ei}(P^{-1})^2 S_{ei}}, \quad (3.10)$$

де P^{-1} – обернена кореляційна матриця;

S_{ei} - усереднений вектор ознак і-го еталона;

arg min – мінімальне значення імовірності.

3.2.5 Кореляційний алгоритм

Проведемо оцінку ефективності застосування розглянутих алгоритмів розпізнавання порівняно з широко розповсюдженим кореляційним алгоритмом [86; 87]. Формула кореляційного алгоритму алгоритму приведена у 3.11.

$$K = \frac{1}{L} \frac{\sum_{L=1}^{L} (S_L - \bar{S}_L)(S_{ei} - \bar{S}_{ei})}{\sigma_s \sigma_{ei}}, \quad (3.11)$$

де L – кількість проведених при розпізнаванні випробувань;

$(S_L - \bar{S}_L)$ - L-й приймаючий та усереднений по P вектори ознаки;

$\bar{S}_{ei}$ - усереднений вектор ознак і-го еталону;

σ_s та σ_{ei} - середнє відхилення різниць $(S_L - \bar{S}_L)$ та $(S_{ei} - \bar{S}_{ei})$.

3.2.6 Вдосконалення системи ідентифікації мережевих атак за рахунок визначення ефективного алгоритму

Для визначення ефективного алгоритму була проведена порівняльна характеристика алгоритмів цифрового спектрального аналізу. Для порівняльної оцінки запропонованих алгоритмів ІМА було проведено моделювання вимірювальних ознак з наступним прийняттям рішення, що має данні ознаки.

Значення ознак розпізнавання (S) не перевищувало восьми. Імовірність правильної ідентифікації (ІПІ) була визначена від 0 до 1. Було здійснене

моделювання для кожного конкретного випадку за допомогою датчика випадкових чисел з рівномірним, нормальним та законом розподілення Лапласа, які зображені на рисунках 3.5, 3.6 та 3.7 відповідно.

На поданих графіках використані наступні позначення обраних алгоритмів розпізнавання:

- МВ –Махалонобісової відстані.
- Б – Байєса.
- К – кореляційний.
- КП – Кейпона.
- ТШ – "тепловий шум".

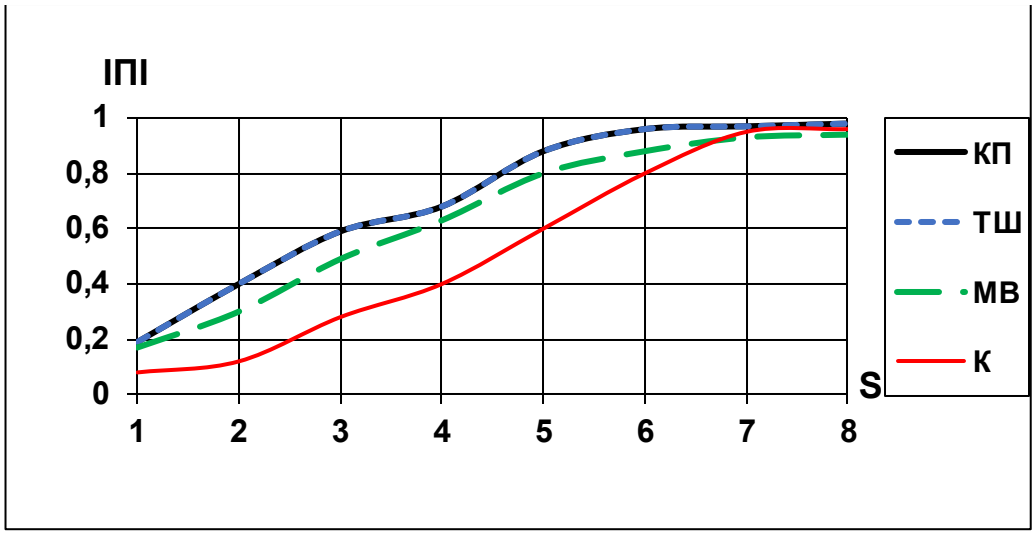

Рисунок 3.9 - Залежність ІПР від кількості ознак розпізнавання для рівномірного закону розподілу

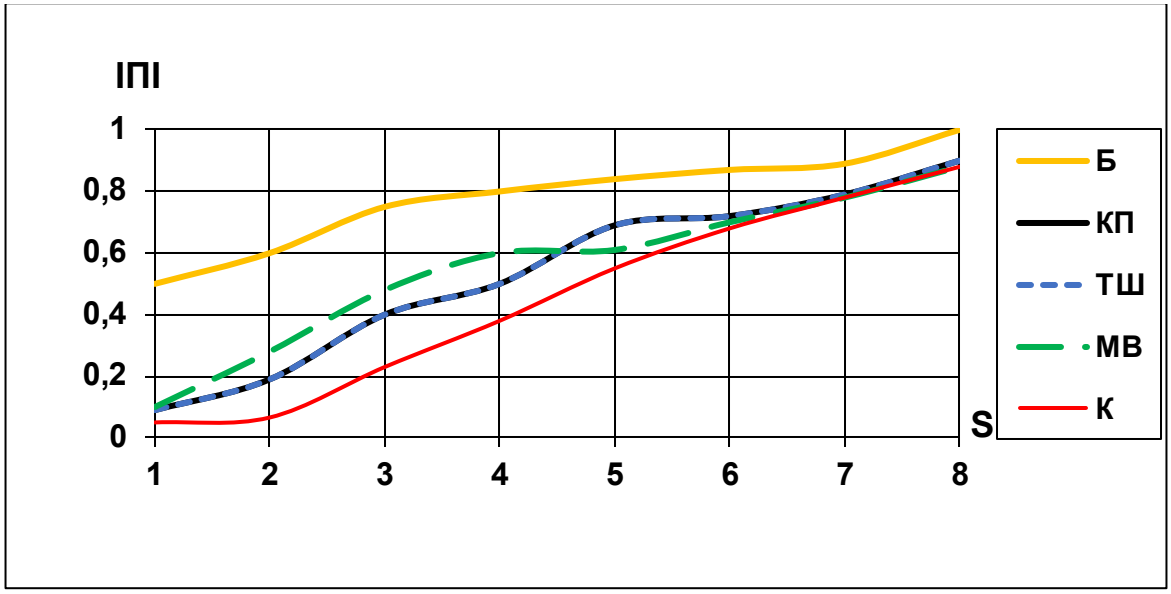

Рисунок 3.10 - Залежність ІПР від кількості ознак розпізнавання для нормального закону розподілу

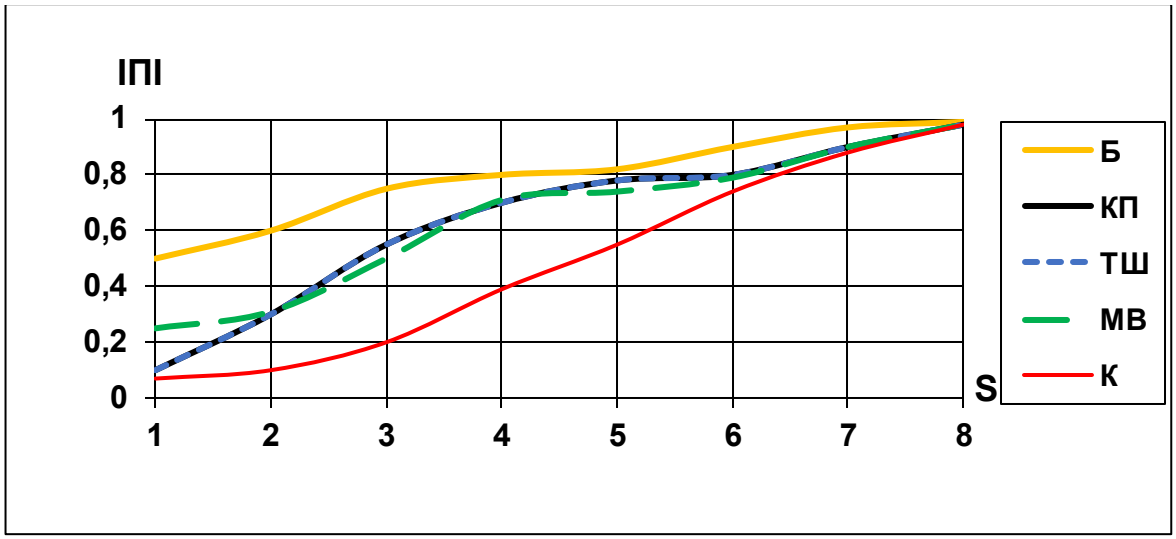

Рисунок 3.11 - Залежність ІПР від кількості ознак розпізнавання для закону розподілу Лапласа

При рівномірному закону розподілу (рис. 3.9) аналіз алгоритму Байеса не проводився в зв'язку с тим, що для його роботи необхідне знання максимум значення функції закону розподілення вектору ознак, який на всій своїй протяжності має однакове значення [85; 86; 87].

Так можемо спостерігати з рисунків 3.9, 3.10, 3.11, що алгоритм Кейпона показує найвищі результати серед інших алгоритмів при різних умовах закону розподілу.

На рисунку 3.12 зображено застосування алгоритму Кейпона.

Рисунок 3.12 – Компоненти системи ідентифікації шкідливих листів

Із отриманих результатів можна стверджувати, що порівнявши вибірку алгоритмів ІМА, найефективнішим виявився алгоритм Кейпона. Застосування цього алгоритму у системі ідентифікації мережевих атак буде ефективним при його взаємодії із підсистемою аналізу даних, який зображений на рисунку 3.8.

Тому застосування цього алгоритму в підсистемі аналізу даних, може вдосконалити систему ідентифікації мережевих атак на величину до 5% [87].

Висновки до розділу 3

В розділі 3 був проведений порівняльна характеристика алгоритмів цифрового спектрального аналізу та був обраний один із запропонованих алгоритмів для вдосконалення системи ідентифікації мережевих атак.

За час розвитку ідентифікації мережевих атак було створено достатньо велику кількість алгоритмів для розпізнавання шкідливого мережевого трафіку. З часом вони вдосконалювалися для того щоб бути ефективними з точки зору

безпеки. Проте важливим чинником у побудові архітектури захищеної мережі, час за, яким здійснюється виявлення ознак є вагомим атрибутом. Так, алгоритми ІМА мають бути не тільки ефективними з точки зору безпеки, але й повинні обробляти велику кількість мережевих даних за одиницю часу.

Серед алгоритмів, які вже використовуються в системах IDS\IPS були розглянуті:

- паралельні обчислення фільтрів Блума;
- алгоритм порівняння шаблонів мережевого процесору;
- алгоритм відповідності шаблонів TCAM;

Проте використання одного з таких алгоритмів на сьогодення не може гарантувати максимальну безпеку у мережі.

Були описані можливі алгоритми для використання їх на системах виявлення і запобігання вторгнень. Кожний з алгоритмів має власні переваги та недоліки серед інших. Атрибутами для порівняння вищевказаних алгоритмів були:

- кількість дій на одиницю часу;
- продуктивність з точки зору безпеки;
- навантаження на систему.

Проте жодний із запропонованих алгоритмів не відповідав вимогам сучасності, а саме ефективності та швидкодії. Одним з рішень було проаналізувати низку алгоритмів ЦСА і порівняти їх. Так у експерименті були запропоновані алгоритми:

- Махалонобісової відстані.
- Байєса.
- Кореляційний.
- Кейпона.
- "Теплового шуму".

Із аналізу отриманих результатів можна стверджувати, що порівнявши вибірку алгоритмів ІМА, найефективнішим виявився алгоритм Кейпона. Тому

застосування цього алгоритму в підсистемі аналізу даних, може підвищити її ефективність ідентифікації мережевих атак на величину до 5% [87].

ВИСНОВКИ

У дипломній роботі було досліджено та проаналізовано методи реалізації сучасних мережевих атак з боку шкідливих листів, досліджено механізми ідентифікації шкідливих листів, був виконаний аналіз існуючих алгоритмів розпізнавання шкідливих листів, виконаний порівняльний аналіз запропонованих алгоритмів для антиспам системи.

На основі досліджень була обрана система ідентифікації мережевих атак, яка спроможна не тільки виявляти, а й запобігати з найбільшою ефективністю. Для досягнення поставленої мети роботи було проведено порівняльний аналіз запропонованих алгоритмів цифрового спектрального аналізу.

За результатами проведення порівняльного аналізу був сформований висновок, що використання одного із запропонованих алгоритмів буде сприяти підвищенню ефективності роботи системи ідентифікації шкідливих листів. Так, як саме завдяки застосуванню запропонованого алгоритма Кейпона можна спостерігати підвищення коефіцієнта розпізнавання при малій кількості ознак.

Практичне значення роботи полягає у підвищенні ефективності роботи системи ідентифікації шкідливих листів, а саме у виборі алгоритму при якому система буде спроможна розпізнати велику кількість ознак за одиницю часу наближеного до реального. Також алгоритм має бути достатньо легких у обчисленні системою, це буде сприяти довготривалій та стабільній роботі системи ідентифікації мережевих атак з боку шкідливих листів. На відповідних графіках були продемонстровані результати роботи алгоритмів цифрового спектрального аналізу. Результати, які отримані в дипломній роботі можуть бути використані в подальшому, як рекомендації до вдосконалення системи ідентифікації шкідливих листів, а саме для підвищення її ефективності.

Поставлені в роботі задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Top 10 most common types of cyber attacks [Електронний ресурс]. – Режим доступу: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/

2) Various types network attacks [Електронний ресурс]. – Режим доступу: https://www.symantec.com/connect/articles/security-11-part-3-various-types-network-attacks

3) Network attack types [Електронний ресурс]. – Режим доступу: https://www.calyptix.com/top-threats/top-7-network-attack-types-2016/

4) Basic network attacks [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/basic-network-attacks/

5) A Role-Based Trusted Network Provides Pervasive Security and Compliance [Електронний ресурс]. – Режим доступу: https://newsroom.cisco.com/feature-content?type=webcontent&articleId=4124873

6) Спам и фишинг в 2018 [Електронний ресурс] Режим доступу: https://securelist.ru/spam-and-phishing-in-2018/93453/

7) Does your PC have a good rep?. [Електронний ресурс] Режим доступу https://www.cnet.com/news/does-your-pc-have-a-good-rep-to-send-e-mail-it-better/

8) Eliminate Spam. [Електронний ресурс] Режим доступу http://awildduck.com/?p=277

9) Desktop Windows Version Market Share Worldwide. [Електронний ресурс] Режим доступу: http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide#monthly-201902-201902-bar

10) Desktop Operating System Market Share Worldwide. [Електронний ресурс] Режим доступу: http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201902-201902-bar

11) Desktop macOS Version Market Share Worldwideю. [Електронний ресурс] Режим доступу: http://gs.statcounter.com/os-version-market-share/macos/desktop/worldwide#monthly-201902-201902-bar

12) Методы автоматической классификации текста. [Електронний ресурс] Режим доступу: http://www.swsys.ru/index.php?page=article&id=425

13) Синтаксический анализ в NLTK. [Електронний ресурс] Режим доступу: https://habr.com/ru/post/340574/

14) NLTK Workbook [Електронний ресурс] Режим доступу: http://www.nltk.org/book/ch06.html

15) Классификация с помощью мешка слов. Руководство [Електронний ресурс] Режим доступу: http://datareview.info/article/klassifikatsiya-tekstov-s-pomoshhyu-meshka-slov-rukovodstvo/

16) Garreta R. Learning scikit-learn: Machine Learning in Python / Raúl Garreta Packt Publishing 2013 108 ст.

17) Richert W. Building Machine Learning Systems with Python / Willi Richert Packt Publishing 2013 290 ст.

18) Lutz M. Learning Python, 5th Edition / Mark Lutz O'Reilly Media 2013 -648 ст.

19) An introduction to machine learning with scikit-learn [Електронний ресурс] Режим доступу: https://scikit-learn.org/stable/tutorial/basic/tutorial.html

20) NIST Special Publication 800-115 [Електронний ресурс] // U.S. Department of Commerce. – 2008. – Режим доступу до ресурсу: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.

21) Web Security Testing Guide Version 4.1 [Електронний ресурс] // OWASP. – 2020. – Режим доступу до ресурсу: https://github.com/OWASP/wstg/releases/download/v4.1/wstg-v4.1.pdf.

22) Adobe Flash Player EOL General Information Page [Електронний ресурс] // Adobe. – 2017. – Режим доступу до ресурсу: https://www.adobe.com/uk/products/flashplayer/end-of-life.html.

23) Markov Chains: How to Train Text Generation to Write Like George R. R. Martin [Електронний ресурс] // KDnuggets. – 2019. – Режим доступу до ресурсу: https://www.kdnuggets.com/2019/11/markov-chains-train-text-generation.html.

24) Іванченко С. В. Методи автоматизованої генерації фішингових повідомлень для тестування на проникнення / С. В. Іванченко, І. В. Стьопочкіна. // Теоретичні і прикладні проблеми фізики, математики та інформатики: матеріали XVIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених. – 2019. – №18. – С. 257–260.

25) Білодід І. К. КОСА / Іван Костянтинович Білодід // Словник української мови в 11 томах / Іван Костянтинович Білодід.. – (Том 4). – С. 304.

26) Bengio Y. Learning Long-Term Dependencies with Gradient Descent is Difficult / Y. Bengio, P. Simard, P. Frascioni. // IEEE Transactions on Neural Networks. – 1994. – №5. – С. 157–166.

27) Olah C. Understanding LSTM Networks [Електронний ресурс] / Christopher Olah // colah's blog. – 2015. – Режим доступу до ресурсу: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

28) Giacaglia G. How Transformers Work [Електронний ресурс] / Giuliano Giacaglia // Towards Data Science. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/transformers-141e32e69591.

29) Інформаційна служба КПІ ім. Ігоря Сікорського [Електронний ресурс] // КПІ ім. Ігоря Сікорського, Канцелярія. – 2004. – Режим доступу до ресурсу: https://document.kpi.ua/.

30) Alammar J. Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention) [Електронний ресурс] / Jay Alammar // jalammar. – 2018. – Режим доступу до ресурсу: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/.

31) Alammar J. The Illustrated Transformer [Електронний ресурс] / Jay Alammar // jalammar. – 2018. – Режим доступу до ресурсу: http://jalammar.github.io/illustrated-transformer/.

32) OpenAI [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/OpenAI#GPT.

33) Generate Unconditional Samples [Електронний ресурс] // GitHub. – 2019. – Режим доступу до ресурсу: https://github.com/openai/gpt-2/blob/master/src/generate_unconditional_samples.py.

34) The Curious Case of Neural Text Degeneration [Електронний ресурс] / [A. Holtzman, J. Buys, L. Du та ін.] // International Conference on Learning Representations. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/1904.09751.

35) Mann B. How to sample from language models [Електронний ресурс] / Ben Mann // Towards Data Science. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277.

36) BERT (language model) [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/BERT_(language_model).

37) Horev R. BERT Explained: State of the art language model for NLP [Електронний ресурс] / Rani Horev // Towards Data Science. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

38) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс] / J.Devlin, M. Chang, K. Lee, K. Toutanova // 2019 – Режим доступу до ресурсу: https://arxiv.org/abs/1810.04805.

39) Understanding XLNet [Електронний ресурс] // Borealis AI. – 2019. – Режим доступу до ресурсу: https://www.borealisai.com/en/blog/understanding-xlnet/.

40) XLNet: Generalized Autoregressive Pretraining for Language Understanding [Електронний ресурс] / [Z. Yang, Z. Dai, Y. Yang та ін.] // arXiv. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/1906.08237.

41) Cloud TPU pricing [Електронний ресурс] // Google Cloud. – 2020. – Режим доступу до ресурсу: https://cloud.google.com/tpu/pricing.

42) TPUs in Colab [Електронний ресурс] // Google Colab. – 2020. – Режим доступу до ресурсу: https://colab.research.google.com/notebooks/tpu.ipynb.

43) Deep learning super sampling [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Deep_learning_super_sampling#Architecture.

44) The Staggering Cost of Training SOTA AI Models [Електронний ресурс] // Synced. – 2019. – Режим доступу до ресурсу: https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/.

45) Hui J. AI Chips: Google TPU [Електронний ресурс] / Jonathan Hui // Medium. – 2020. – Режим доступу до ресурсу: https://jonathan-hui.medium.com/ai-chips-tpu-3fa0b2451a2d.

46) Sharir O. The Cost of Training NLP Models: A Concise Overview [Електронний ресурс] / O. Sharir, B. Peleg, Y. Shoham // arXiv. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/2004.08900.

47) Mooney A. How Much Does Custom Software Cost? [Електронний ресурс] / Ann Mooney // Soltech. – 2020. – Режим доступу до ресурсу: https://soltech.net/how-much-does-custom-software-cost/.

48) Joshi N. How to fine-tune your artificial intelligence algorithms [Електронний ресурс] / Naveen Joshi // Allerin. – 2020. – Режим доступу до ресурсу: https://www.allerin.com/blog/how-to-fine-tune-your-artificial-intelligence-algorithms.

49) Shvedova M. The General Regionally Annotated Corpus of Ukrainian (GRAC, uacorpus.org): Architecture and Functionality [Електронний ресурс] /

Maria Shvedova // Computational Linguistics and Intelligent Systems. – 2020. – Режим доступу до ресурсу: http://ceur-ws.org/Vol-2604/paper36.pdf.

50) Full-text corpus data [Електронний ресурс] // English-Corpora.org. – 2014. – Режим доступу до ресурсу: https://www.corpusdata.org/wikipedia.asp.

51) Kobayashi S. Crawl BookCorpus [Електронний ресурс] / Sosuke Kobayashi // GitHub. – 2018. – Режим доступу до ресурсу: https://github.com/soskek/bookcorpus.

52) twitter data [Електронний ресурс] // data.world. – 2020. – Режим доступу до ресурсу: https://data.world/datasets/twitter.

53) Leskovec J. 476 million Twitter tweets [Електронний ресурс] / Jure Leskovec // Stanford Network Analysis Project. – 2011. – Режим доступу до ресурсу: https://snap.stanford.edu/data/twitter7.html.

54) Enron Email Dataset [Електронний ресурс] // Carnegie Mellon University School of Computer Science. – 2015. – Режим доступу до ресурсу: https://www.cs.cmu.edu/~./enron/.

55) Penetration Testing Companies and Vendor List [Електронний ресурс] // HighBit Security. – 2019. – Режим доступу до ресурсу: https://www.highbitsecurity.com/penetration-testing-company-list.pdf.

56) France Pentesting & Security Assessments [Електронний ресурс] // Cyber Security Intelligence. – 2020. – Режим доступу до ресурсу: https://www.cybersecurityintelligence.com/category/pentesting-and-security-assessments/location/france/.

57) Rosset C. Turing-NLG: A 17-billion-parameter language model by Microsoft [Електронний ресурс] / Corby Rosset // Microsoft Research Blog. – 2020. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

58) Kitaev N. Reformer: The Efficient Transformer [Електронний ресурс] / N. Kitaev, L. Kaiser, A. Levskaya // OpenReview.net. – 2020. – Режим доступу до ресурсу: https://openreview.net/forum?id=rkgNKkHtvB.

59) SenseBERT: Driving Some Sense into BERT [Електронний ресурс] / [Y. Levine, B. Lenz, O. Dagan та ін.] // Association for Computational Linguistics. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/1908.05646.

60) Vilela, Douglas W. F. L.; Lotufo, Anna Diva P.; Santos, Carlos R. Fuzzy ARTMAP Neural Network IDS Evaluation applied for real IEEE 802.11w data base. // 2018 International Joint Conference on Neural Networks (IJCNN), 2018. – P. 124-143.

61) ISO/IEC 27039:2015 — Information technology — Security techniques — Selection, deployment and operation of intrusion detection and prevention systems (IDPS)

62) PCI DSS Quick Reference Guide [Електронний ресурс]. – Режим доступу: https://www.pcisecuritystandards.org/documents/PCI%20SSC%20Quick%20Reference%20Guide.pdf

63) OWASP Intelligent Intrusion Detection System [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/OWASP_Intelligent_Intrusion_Detection_System

64) Evolution of intrusion detection system [Електронний ресурс]. – Режим доступу: https://www.secureworks.com/blog/the-evolution-of-intrusion-detection-prevention

65) Network intrusion detection system [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Intrusion_detection_system#Network_intrusion_detection_systems

66) Network security [Електронний ресурс]. – Режим доступу: https://www.cisco.com/c/en/us/products/security/what-is-network-security.html

67) Network behavior anomaly detection [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Network_behavior_anomaly_detection

68) Обзор IPS/IDS [Електронний ресурс]. – Режим доступу: https://xakep.ru/2012/10/29/ids-ips/

69) Yi Wang, Zhuyun Qi, Huichen Dai, Hao Wu, Kai Lei, Bin Liu, "Statistical Optimal Hash-Based Longest Prefix Match" // Architectures for Networking and Communications Systems (ANCS) 2017 ACM/IEEE Symposium on, pp. 153-164, 2017.

70) Fang Yu, R.H. Katz, T.V. Lakshman, "Gigabit rate packet pattern-matching using TCAM", Network Protocols 2004. ICNP 2004. // Proceedings of the 12th IEEE International Conference on, pp. 174-183, 2004.

71) Xuejuan Li, Qiaoyan Wen, "A fast multi-pattern matching algorithm for anti-virus scanning" // Broadband Network and Multimedia Technology (IC-BNMT) 2011 4th IEEE International Conference on, pp. 42-45, 2011.

72) S. Dharmapurikar ; P. Krishnamurthy ; D.E. Taylor, Longest prefix matching using bloom filters // IEEE/ACM Transactions on Networking (Volume: 14 , Issue: 2 , April 2006, p. 397-409.

73) Zhiping Cai, Zhijun Wang, Kai Zheng, Jiannong Cao, "A Distributed TCAM Coprocessor Architecture for Integrated Longest Prefix Matching Policy Filtering and Content Filtering" // Computers IEEE Transactions on, vol. 62, no. 3, pp. 417-427, 2013.

74) Bezroukov, N., "Architectural Issues of Intrusion Detection Infrastructure in Large Enterprises" // Softpanorama Bulletin Vol. 17, No. 3 – 2010. – P. 3-17.

75) Neminath Hubballi, Santosh Biswas, Sukumar Nandi, "Sequencegram: n-gram modeling of system calls for program based anomaly detection", Communication Systems and Networks // Third International Conference on, pp. 1-10, 2011.

76) Марпл-младший С.Л. Цифровой спектральный анализ и его приложения // Москва: Мир, 1990 г. – 265 с.

77) Теорема Байєса [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki Теорема_Байеса

78) Відстань Махаланобіса [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/ Расстояние_Махаланобиса

79) ЕВОЛЮЦІЯ СПАМУ ТА АНТИСПАМЕРСЬКИХ ПРОГРАМ [Електронний ресурс]. – Режим доступу: ela.kpi.ua/bitstream/123456789/28524/1/Ivankov_bakalavr.docx.docx

80) Тепловой шум [Електронний ресурс]. – Режим доступу: http://sci.sernam.ru/book_nps.php?id=10

81) Исследование свойств алгоритма Кейона в задачах пеленгации [Електронний ресурс]. – Режим доступу: http://www.ipo.spb.ru/journal/content/1324/

82) Атака “грубой силой” [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki Атака_«грубой_силой»

83) Наконечний В.С. Аналіз ефективності та можливості застосування сучасних методів розпізнавання об’єктів радіолокаційного моніторингу. // Науково-технічний журнал Зв'язок. - 2014. - №5. - С. 52-56.

84) В.С. Наконечний, С.В. Толюпа, І.Р. Пархомей, Н.В. Цьопа. Експериментальне дослідження надрозрізрювальних методів спектрального аналізу для задач пеленгації. Адаптивні системи автоматичного управління // Міжвідомчий науково-технічний збірник. — Київ: Національний технічний університет України “Київський політехнічний інститут”. – 2015. – Вип. 2(27). – с. 88-94.

85) Вялкова В.І. Підвищення ефективносі розпізнавання мережевих атак за рахунок використання програмно-апаратної системи IPS / Вялкова В.І., Войтенко І.Ю. // Інформаційні технології та взаємодії : V Міжнародна науково-практична конференція, 20 - 21 листопада 2018 року : тези доп. – К.: Київський нац. ун-т імені Тараса Шевченка, 2018. – С. 260-262

86) Толюпа С.В. 1, Наконечный В.С 2., Вялкова, В.И. 3, Войтенко И.Ю. Повышение эффективности выявления сетевых атак за счет алгоритмов распознавания вредоносного трафика // Scientific and Practical Cyber Security Journal (SPCSJ) 3(1): 58-65 ISSN 2587-4667 Scientific Cyber Security Association (SCSA) – 2019. – P. 58-65.

87) Наконечний В.С., Вялкова В.І. Підвищення ефективності виявлення мережевих атак за рахунок алгоритмів виявлення шкідливого трафіку / Наконечний В.С., Вялкова В.І., Войтенко І.Ю. // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: II Міжнародна науково-практична конференція, 11 –12

квітня 2019 року : тези доп. – К.: Київський нац. ун-т імені Тараса Шевченка, 2019. – С. 175-178.