МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри

_____ І.А. Жуков
(підпис)

« ____ » _____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: « Система керування доступом до розподілених веб-ресурсів »

Виконавець: студентка групи КС-231, Павленко Синтія Сергіївна
(студент, група, прізвище, ім'я, по батькові)

Керівник: кандидат технічних наук, доцент, Гузій Микола Миколайович
(наукова ступінь, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: _____ Малярчук В.О.
(підпис) (ПІБ)

Засвідчую, що у дипломній роботі немає запозичень праць інших авторів без відповідних посилань
Студент _____ Павленко С.С.
(підпис) (ПІБ)

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ І.А. Жуков
(підпис)

« ____ » _____ 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи

Павленко Синтії Сергіївни
(прізвище, ім'я, по батькові)

1. Тема роботи: Система керування доступом до розподілених веб-ресурсів

затверджена наказом ректора від « 25 » вересня 2020 року № 1793/ст.
2. Термін виконання роботи: з 05.10.2020 р. по 30.12.2020 р.
3. Вихідні дані до роботи: вимоги до системи керування доступом до розподілених веб-ресурсів

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
вступ, аналіз технологій та процесів розподілених веб-ресурсів, механізми керування доступом до розподілених веб-ресурсів, методи та засоби керування доступом до розподілених веб-ресурсів, розробка системи керування доступом до розподілених веб-ресурсів, висновки
5. Перелік обов'язкового графічного матеріалу:
Презентація *PowerPoint*

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1	Отримати завдання дипломної роботи	05.10.2020	
2	Скласти план дипломної роботи	10.10.2020	
3	Ознайомитись з літературними джерелами	15.10.2020	
4	Провести аналіз технологій та процесів розподілених веб-ресурсів	24.10.2020	
5	Написати перший розділ	29.11.2020	
6	Провести дослідження механізмів керування розподіленими веб-ресурсами	05.11.2020	
7	Написати другий розділ	10.11.2020	
8	Провести дослідження інструментальних засобів керування доступом до розподілених веб-ресурсів	16.11.2020	
9	Написати третій розділ	23.11.2020	
10	Розробити систему керування доступом до розподілених веб-ресурсів	27.11.2020	
11	Написати четвертий розділ	02.12.2020	
12	Оформити пояснювальну записку	07.12.2020	

7. Дата видачі завдання: « 05 » жовтня 2020 р.

Керівник дипломної роботи: _____ Гузій М.М.
(підпис) (П.І.Б)

Завдання прийняла до виконання: _____ Павленко С.С.
(підпис випускника) (П.І.Б)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Система керування доступом до розподілених веб-ресурсів”: 85 сторінок, 35 рисунків, 33 використаних джерел.

ВЕБ-РЕСУРС, КЕРІВНИЙ ВПЛИВ, *HTTP*-СЕСІЯ, ВЕБ-ФІЛЬТР, АНАЛІЗАТОР ЗАГРОЗ, *C++*, *SQLITE*, ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ПРОКСІ-СЕРВЕР.

Мета дипломної роботи – дослідити механізми доступу до розподілених веб-ресурсів, визначити можливі керівні впливи на процес доступу до розподілених веб-ресурсів, сформувати функціональні вимоги системи керування доступом до розподілених веб-ресурсів, проаналізувати та обґрунтувати вибір технологій для розробки серверних додатків та на їх основі створити систему керування доступом до розподілених веб-ресурсів.

Об’єкт дослідження: розподілені інформаційні ресурси в мережі Інтернет.

Предмет дослідження: система керування доступом до розподілених веб-ресурсів.

Методи дослідження: евристичний аналіз, сигнатурний аналіз, системний аналіз, методи програмної інженерії, клієнт-серверна архітектура, *UML*-проектування.

Отримані результати та наукова новизна: досліджені механізми доступу до розподілених веб-ресурсів; узагальнені види інтернет-загроз та зазначена важливість захисту від них; набули подальшого розвитку концепції аналізу та виявлення інтернет-загроз у веб-трафіку; запропонована концепція використання прямого *HTTP*-проксі серверу для аналізу веб-трафіку та виявлення інтернет-загроз; обрані та обґрунтовані технології для створення та створена система керування доступом до розподілених веб-ресурсів.

Рекомендації щодо використання результатів: впровадження створеної системи керування доступом до розподілених веб-ресурсів у роботу та її використання у локальних та глобальних мережах.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

CWE	–	*Common Weakness Enumeration*
DDOS	–	*Distributed Denial-of-Service Attack*
DNS	–	*Domain Name System*
GDPR	–	*General Data Protection Regulation*
HTTP	–	*HyperText Transfer Protocol*
IDE	–	*Integrated Development Environment*
RFC	–	*Request for Comments*
SSL	–	*Sockets Secure Layer*
TCP	–	*Transmission Control Protocol*
TLS	–	*Transport Layer Security*
URI	–	*Uniform Resource Identifier*
URL	–	*Uniform Resource Locator*

ВСТУП

Проблеми пошуку та застосування засобів для керування доступом до Інтернет-ресурсів стають одними із ключових у зв'язку з ростом інтенсивності користування Інтернетом, перенесенням інформаційних просторів у онлайн, розширенням каналів зв'язку.

Кількість користувачі в всесвітньої мережі Інтернет зростає, створюються нові методи і технології доступу до неї. Підвищений попит на інформаційні ресурси спричинив виникнення пропозиції. Кількість доступних веб-ресурсів зростає. Створюються нові веб-сайти та сервіси.

Такий стрімкий розвиток призвів до необхідності створення нових інженерних інструментів для спрощення розгортання веб-серверів, оптимізації потоків даних та шляхів їх проходження, керування доступом до веб-ресурсів тощо. Збільшення навантаження та зростання кількості шкідливого програмного забезпечення (ПЗ) у мережі – це очевидний результат вищеописаних процесів.

Для вирішення цих проблем низка світових компаній створює та підтримує антивірусне ПЗ, що задовольняє попит користувачів у захисті від будь-яких інтернет-загроз. (*AVZ*, *KasperskyLabs*, *ESET* тощо). Більшість механізмів боротьби їхнього ПЗ полягають у виявленні загроз після попадання до кінцевої машини користувача. Аналіз здійснюється також апаратними ресурсами кінцевого користувача. Інші ж підходи полягають у аналізі інтернет-загроз на етапах перед попаданням його на комп'ютер користувача.

Отже, обрана тема дипломної роботи є актуальною з точки зору сучасних тенденцій, вимагає проведення дослідження у сфері керування та аналізу веб-трафіку, а відповідно потребує значних інженерних навичок та умінь в області мережевого та системного програмування.

Проблематика та напрямки роботи з веб-трафіком:

1. Протоколи *HTTP/HTTPS*.
2. Шифрування веб-трафіку.
3. Стиснення даних.

4. Захист від інтернет-загроз.

5. Захист від *DDOS*-атак.

До основної проблематики дипломної роботи можливо віднести необхідність у розробці системи керування доступом до розподілених веб-ресурсів, дослідженні можливих керівних впливів на цей процес, аналізі технологій доступу до веб-ресурсів, досліджені форматів їх пересилання, розуміння роботи протоколів *HTTP/HTTPS* та *TCP*, вирішенню питань по аналізу веб-трафіку на наявність інтернет-загроз, оптимізації швидкодії створюваної системи, навичок у проектуванні та програмуванні серверних додатків.

Метою дипломної роботи є дослідження механізмів доступу до розподілених веб-ресурсів, оцінка можливих загроз при доступі до розподілених веб-ресурсів та з'ясування методів їх виявлення, аналіз та обґрунтування вибору технологій для розробки серверних додатків та на їх основі створення системи керування доступом до розподілених веб-ресурсів.

Об'єкт дослідження є розподілені інформаційні ресурси в мережі Інтернет.

Предметом дослідження є система керування доступом до розподілених веб-ресурсів: компонентний склад та взаємозв'язки між компонентами, механізми передачі даних.

У ході виконання дипломної роботи були використані наступні **методи**: евристичний аналіз, сигнатурний аналіз, системний аналіз, методи програмної інженерії, клієнт-серверна архітектура, *UML*-проектування.

Використання вищеназваних методів обумовлюється необхідністю проведення досліджень методів виявлення інтернет-загроз, дослідженні технологій проксі-серверів та використання їх технологій у роботі по створенню системи керування доступом до розподілених веб-ресурсів, а відповідно і проектуванню та програмуванню цієї системи.

Наукова новизна отриманих результатів:

1. Досліджені та узагальнені керівні впливи на процес доступу до розподілених веб-ресурсів.

2. Узагальнені методи виявлення інтернет-загроз у комп'ютерних мережах.

3. Досліджений вплив технологій шифрування та стискання даних на процес аналізу веб-трафіку.

4. На основі запропонованої математичної моделі сигнатурного аналізу мережевого трафіку розроблено алгоритм сигнатурного аналізу веб-трафіку на наявність інтернет-загроз.

5. Створена система керування доступом до розподілених веб-ресурсів, яка встановлює правила доступу користувача до веб-ресурсів, аналізує вхідний трафік на наявність інтернет-загроз та попереджує потрапляння шкідливого трафіку до кінцевого користувача.

Згідно мети дипломної роботи необхідно вирішити наступні **задачі**:

1. Розкрити поняття “веб-ресурс”: розглянути суть поняття, описати основні механізми обміну веб-ресурсами, описати технології шифрування та стискання при обробці веб-ресурсів.

2. Дослідити можливі керівні впливи на процес керування доступом до розподілених веб-ресурсів та обрати конкретні керівні впливи для дослідження.

3. Дослідити способи та методи виявлення інтернет-загроз у веб-трафіку.

4. Провести дослідження технологій проксі-серверів з метою їх використання у розроблюваній системі.

5. Визначити функціональні вимоги до розроблюваної системи.

6. Обґрунтувати вибір технологій для розробки.

7. Спроектувати систему керування доступом до розподілених веб-ресурсів: розділити систему на компоненти та встановити зв’язки між ними, створити *UML*-діаграми класів та схеми БД.

8. Розробити програмне забезпечення системи керування доступом до розподілених веб-ресурсів, надати рекомендації щодо її розгортання та використання, представити результати роботи системи.

Практичне значення отриманих результатів полягає у проведеному дослідженні технологій доступу до розподілених веб-ресурсів, здійсненому аналізі підходів до виявлення інтернет-загроз у мережевому трафіку, створеній системі керування доступом до розподілених веб-ресурсів.

Технології, що були задіяні при виконанні дипломної роботи наступні: мова програмування *C++*, фреймворк для розширення функціоналу стандартної бібліотеки *C++ – Qt Framework, SQLite3* – основою базою даних для зберігання інформації. Використані технології забезпечили наступне:

1. Швидку розробку та зручне відлагодження системи у процесі її створення.

2. Повну кросплатформеність створеного ПЗ: скомпілювати та використати його можливо на будь-якій ОС та платформі, за наявності компілятора *C++*.

3. Оптимізовану швидкість роботи за рахунок використання низькорівневої мови.

4. Можливості у збереженні масивної кількості даних локально – за рахунок технології БД.

Додатково до вищезазначеного необхідно зазначити технології проксі-серверів, оскільки вони були описані у роботі та на основі їх механізмів встановлення з'єднань базується комунікаційна частина системи, а саме – на правилах та механізмах *HTTP*-проксі.

У результаті створена система керування доступом до розподілених веб-ресурсів працює за призначенням: блокує доступ до визначених веб-ресурсів, аналізує веб-трафік на наявність інтернет-загроз, при цьому аналізує ~65536 байт даних на 100000 сигнатур вірусів за ~500мс. Результати роботи системи є задовільними, система вносить лише невеликі затримки у робочі процеси користувачів.

Створену систему керування доступом до розподілених веб-ресурсами можливо впроваджувати у робочі процеси та використовувати її за призначенням для вирішення реальних задач.

Важливо зазначити, що створена система відповідає мінімальним вимогам швидкодії. Для подальшого розвитку рекомендується застосування механізмів розпаралелення виконуваної роботи по декільком ядрам з метою прискорення аналізу веб-трафіку на наявність інтернет-загроз.

Варто зазначити, що система базується лише на одному варіанті для виявлення загроз – сигнатурному аналізі. Припущенням подальшого розвитку у цій області є

застосування більш досконалих алгоритмів виявлення інтернет-загроз, наприклад методів евристичного аналізу для знаходження загроз поліморфічного типу, або ж методів, що базуються на інтелектуальному аналізі – на базі штучних нейронних систем. Також можливим варіантом є використання декількох методів одночасно, проте це є предметом подальших досліджень.

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЙ ТА ПРОЦЕСІВ РОЗПОДІЛЕНИХ ВЕБ-РЕСУРСІВ

1.1. Поняття веб-ресурсу

Веб-ресурс – це інформація, яка доступна для завантаження з всесвітньої мережі Інтернет за допомогою об’єднаних апаратних та програмних засобів, та яка унікально ідентифікується за допомогою уніфікованого локатора ресурсу (*URL*). Веб-ресурсом може бути текст, мультимедіа, графіка, веб-сторінки, документи тощо.

Веб-ресурси передаються через мережу згідно до правил протоколу *HTTP*, специфікаціями якого є: *HTTP*/1.0, *HTTP*/1.1, *HTTP*/2.0. Специфікації повноцінно описані документами *RFC1945*, *RFC7231*, *RFC7540* [1-3].

Зазвичай веб-ресурси створюються у процесі розробки веб-додатків та веб-сайтів відповідно до функціональних вимог останніх.

Кожен веб-ресурс має наступні ознаки:

1. Унікально ідентифікується у межах домену за допомогою унікального ідентифікатора ресурсу (*URI*).

2. Доступний за посиланням у вигляді унікального локатора ресурсу (*URL*).

3. Має назву та тип.

Робота з веб-ресурсами з точки зору будь-якої програмної системи – це комплексний процес, у якому задіяні наступні учасники (рис. 1.1):

1. Канал зв’язку з мережею Інтернет.

2. Клієнтська програма, що надсилає запити на отримання веб-ресурсів.

3. Протоколи обміну даними (*HTTP*), що встановлюють правила зв’язку, які зрозумілі усім сторонам обміну даними (веб-ресурсами).

4. Веб-сервер, який надає доступ до веб-ресурсу за запитом по *URL*.

5. *DNS*-сервер, що призначений для транслювання доменної частини *URL* з форми рядкової у форму *IP* (опціонально, за вимогою).

6. Проксі-сервери, мережеві фільтри та екрани – для керування доступом до веб-ресурсів, встановлення шифрованих з'єднань, стиснення даних, аналізу трафіку та запитів тощо (опціонально).

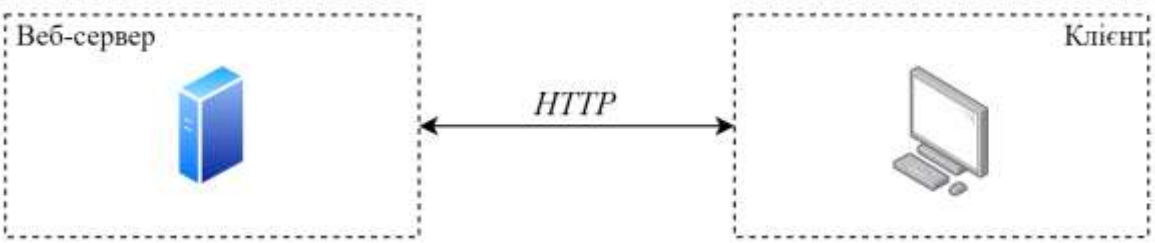

Рис.1.1. Спрощена схема учасників обміну веб-ресурсами

Через масове використання вебу та стандартизацію протоколу *HTTP* операційні системи та мови програмування інкапсулюють більшу частину деталей роботи з веб-ресурсами у прості високорівневі операції, приховуючи інші низькорівневі процеси: встановлення *TCP* з'єднань з веб-серверами, транслювання доменних імен за допомогою *DNS*-серверів, створення обробників *HTTP*-запитів та інші.

1.2. Механізми обміну веб-ресурсами

Запит веб-ресурсу – це сформоване за правилами протоколу *HTTP* повідомлення з метою отримання даних конкретного веб-ресурсу. Будь-який запит містить:

1. Метод запиту, *URI* ресурсу, використовувану версію *HTTP* протоколу.
2. *HTTP*-заголовки.
3. Тіло запиту.

Основні методи запиту: *GET, POST, PUT, HEAD, CONNECT, OPTIONS* тощо. Метод не обмежує тип ресурсу, а лише визначає інтерфейс, що використовується для взаємодії з ним.

Коли клієнт створює *HTTP*-запит, він надсилає цільовий *URI* в одній із форм (*request-target*):

1. *origin-form*.
2. *absolute-form*.

3. *authority-form.*

4. *asterisk-form*.

Origin-form використовується при прямих запитах до цільового серверу, виключаючи *CONNECT* та *OPTIONS* запити. Клієнт повинен зазначити лише абсолютний шлях та, за необхідністю, додаткові параметри *query* цільового *URI*. Також клієнт повинен надіслати заголовок *Host*. Приклад запиту:

GET /priklad?q=now HTTP/1.1

Host: www.priklad.com

Таким чином робота з представленням будь-якого веб-ресурсу формується за правилом:

<HTTP METHOD> <absolute_path?query> <HTTP version>

<Host header>: <host>

Absolute-form використовується при запитах до проксі, виключаючи *CONNECT* та *OPTIONS* запити. Клієнт повинен зазначити цільовий *URI* в *absolute-form*.

absolute-form = absolute-URI

Проксі-сервер обслуговує запит клієнта, надаючи відповідь з кеш-пам'яті, при її наявності, або здійснює такий же запит від імені клієнта до наступного проксі-серверу чи безпосередньо на сервер, що зазначений в *request-target*. Приклад *absolute-form*:

GET HTTP://www.priklad.org/pub/WWW/TheProject.html HTTP/1.1

Authority-form використовується лише для запитів типу *CONNECT*.

authority-form = authority

Створюючи запит *CONNECT* для встановлення тунелю через один або більше проксі-серверів, клієнт повинен надіслати лише *authority* компонент цільового *URI* (за винятком будь-якої інформації про користувача та розмежувача "@") як *request-target*. Приклад:

CONNECT www.priklad.com:80 HTTP/1.1.

Asterisk-form використовується лише для запиту типу *OPTIONS.*

asterisk-form = ""*

Коли клієнт бажає отримати відповідь на запит *OPTIONS* стосовно усього сервера, а не тільки стосовно конкретного іменованого ресурсу цього серверу, то клієнт повинен надіслати “*” (%x2A) як *request-target*. Наприклад:

*OPTIONS * HTTP/1.1*

Якщо проксі-сервер отримує запит *OPTIONS* з *absolute-form* у якій *URI* має пустий шлях та відсутні компоненти запиту, то останній проксі у ланцюгу запитів повинен надіслати *request-target* із позначкою “*”, коли він переадресовує запит на вказаний веб-сервер. Наприклад, запит *OPTIONS http*://www.priklad.org:8001 *HTTP/1.1* був би переадресований останнім проксі-сервером як *OPTIONS * HTTP*/1.1 *Host: www.priklad.org:8001* після зв’язку з портом 8001 хосту “*www.priklad.org*”. Після отримання запиту сервер відновлює ефективний *URI* запиту для цільового ресурсу.

Окрему увагу варто приділити *HTTP*-заголовкам. *HTTP*-заголовок – це окрема частина запиту, що містить додаткові метадані, які визначають додаткові параметри цього запиту. У теорії кількість *HTTP*-заголовків – необмежена, що створює додаткові загрози переповнення буферу оперативної пам’яті веб-серверу (рис. 1.2).

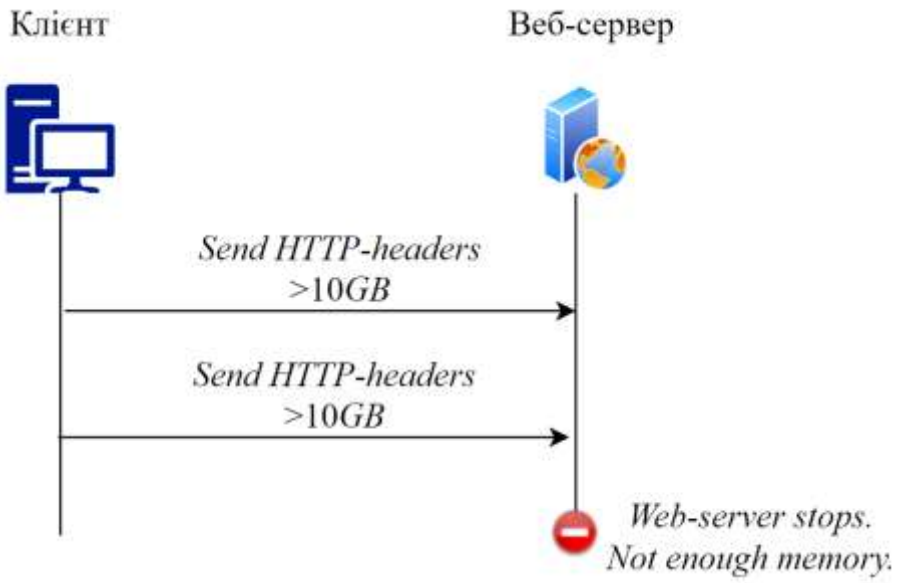

Рис.1.2. Переповнення буферу оперативної пам’яті веб-серверу

Зазвичай дані проблеми усувають шляхом обмеження кількості заголовків, або ж загальним обмеженням доступної для них оперативної пам’яті. Цими питаннями повинен займатись зворотній проксі-сервер або безпосередньо сам веб-сервер. Під

час обробки чергового *HTTP*-запиту відслідковуємо завеликі заголовки та одразу у відповідь надсилаємо код відповіді 431 – *Request Header Fields Too Large*.

Розглянемо наступні *HTTP*-заголовки:

1. *Content-Type*.
2. *Content-Encoding*.
3. *Content-Length.*
4. *Cache-Control*.
5. *Connection*.

Заголовок *Content-Type* вказує клієнту про тип ресурсу. Можливі випадки, коли вказаний тип ресурсу не відповідає дійсності, тому за замовченням більшість клієнтів аналізують повернений веб-ресурс та намагаються встановити його тип самостійно. При відсутності заголовку та неможливості проаналізувати його тип дані вважаються узагальненим байтовим потоком даним (*application/octet-stream*).

Content-Type = media-type

Неправильне використання даного заголовку може привести до застосування вразливостей описаних у *CWE-16: Configuration* та *CWE-436: Interpretation Conflict*.

Заголовок *Content-Length* вказує про довжину тіла повідомлення (самих даних) у байтах. Веб-сервери мають певні обмеження на значення цього заголовка. Обмеження встановлюються через необмежену кількість оперативної пам’яті на сервері, а спроби надіслати запити з великою кількістю даних можуть спричинити його аварійну зупинку через її нестачу. Заголовок можливо взагалі не вказувати, тоді веб-сервер буде змушений самостійно обчислювати довжину тіла. Під час цього процесу веб-сервер має постійно відслідковувати довжину тіла запиту та при її перевищені – одразу перервати з’єднання. Невірною є спроба спочатку завантажити увесь запит, а потім вже обчислювати його довжину, через вищезазначену ситуацію.

Content-Length = <decimal number of octets>

Заголовки *Cache-Control* зберігає інформацію про необхідність кешування запитів та відповідей протягом певного проміжку часу.

Параметрами кешування є:

1. *public* – відповідь може бути закешована будь-де.

2. *private* – відповідь може бути збережена до кешу лише браузером.

3. *no-cache* – відповідь може бути закешована будь-де, проте перед повторним використанням необхідно отримати підтвердження її дійсності від веб-серверу.

4. *no-store* – відповідь не може бути збережена у кеші взагалі.

Параметром часу кешування є: *max-age=<seconds>* – проміжок часу у секундах протягом якого відповідь вважається дійсною.

Заголовок *Connection* визначає стан з'єднання після запиту (рис. 1.3). Його параметри:

1. *keep-alive* – параметр, що визначає використання поточного з'єднання для подальших запитів.

2. *close* – значення за замовченням, означає, що після обробки запиту з'єднання необхідно закрити.

Постійно відкрите з'єднання скорочує час очікування клієнта, оскільки його повторне відкриття вимагає додаткових часових, мережевих та апаратних ресурсів.

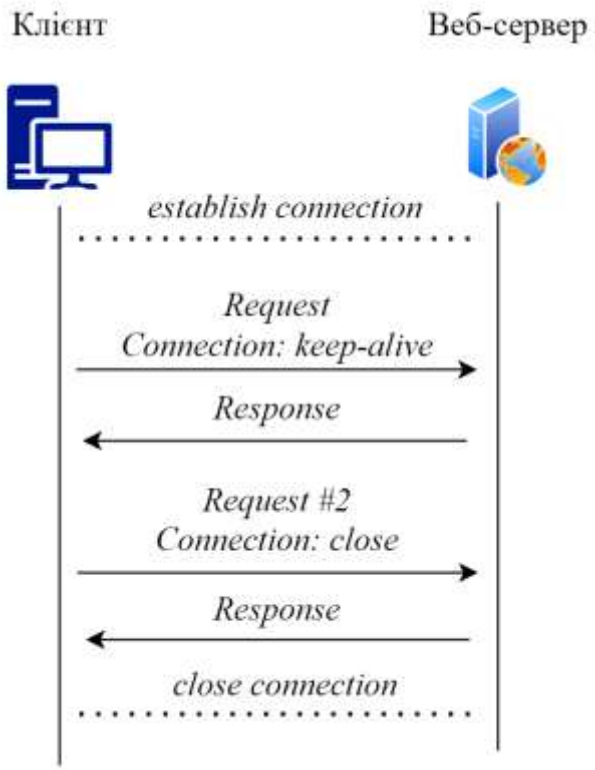

Рис.1.3. Процес встановлення з'єднання та обміну даними

1.3. Механізми шифрування даних при обміні веб-ресурсами

Однією з найважливіших складових процесу обміну веб-ресурсами є створення довіреного середовища, де потенційні користувачі почуваються впевнено у здійсненні будь-яких дій зі своїми персональними даними.

Сертифікати *SSL* створюють основу довіри шляхом встановлення безпечного з'єднання. Використання протоколу *HTTPS*, а конкретно протоколів *SSL* та його нащадку *TLS*, забезпечують шифрування та безпеку даних користувачів при їх пересиланні по мережі у межах веб-сесій.

Спочатку дані в Інтернеті передавалися у відкритому вигляді. Будь-хто міг прочитати їх перехопивши повідомлення. Наприклад, якщо користувач відвідав веб-сайт покупок, зробив замовлення та ввів номер своєї кредитної картки, то у подальшому цей номер кредитної картки надсилався на веб-сервер у відкритому вигляді. Це відбувалося у часи протоколу *HTTP*, який не встановлював правила захисту даних при їх пересиланні.

HTTPS був створений для виправлення проблем захисту конфіденційності користувачів. Протокол *HTTPS* за час існування впровадив використання наступних протоколів шифрування даних: *SSLv2*, *SSLv3*, *TLSv1*, *TLSv1.1*, *TLSv1.2*, *TLSv1.3*.

Протоколи *SSL* та *TLS* виконують наступні функції:

1. Автентифікація та верифікація. Сертифікат *TLS*/*SSL* містить інформацію про автентичність інформації про користувача, компанію або веб-сайт.

2. Шифрування даних. Сертифікат *TLS*/*SSL* забезпечує реалізацію механізму шифрування даних.

Сертифікат *TLS*/*SSL* – це своєрідний цифровий паспорт, що видається спеціальними сертифікаційними центрами (*CA*). У сфері вебу видача сертифікату підпорядковується певним процедурам, основна мета яких впевнитись, що видачу сертифікату запросив справжній власник домену. До таких процедур відносяться:

1. Створення *.txt DNS* запису з згенерованим заздалегідь токеном *CA*.

2. Завантаження згенерованого *CA* файлу з серверу, на який вказує доменний *AAA* запис.

Якщо у першому випадку необхідні навички роботи з системами керування *DNS*-записами, то другий випадок вимагає знань певних інструментів, що забезпечують функціонал *HTTP*-серверів (наприклад проксі по типу *NGINX*, або ж будь-які інші веб-сервери). Тим не менш, після впровадження сертифікату у роботу користувачі веб-сайтом зможуть запевнитись у надійності та автентичності з'єднання з сервером, а конкретну інформацію про це повинен надати їх веб-браузер.

Існують різні типи сертифікатів. Один сертифікат може застосовуватися до одного веб-сайту або кількох веб-сайтів, залежно від типу:

1. *Single-domain*. Сертифікат, що застосовується для одного домену.

2. *Wildcard*. Сертифікат, що застосовується для одного домену. Може включати усі або декілька субдоменів цього домену.

3. *Multi-domain*. Багатодоменний сертифікат. Може включати декілька непов'язаних доменів.

Детальніше розглянемо протоколи *SSL* та *TLS*. *SSL* є безпосереднім попередником протоколу *TLS*.

Використовуваний метод шифрування даних у обох протоколах називається асиметричною криптографією. Суть асиметричної криптографії випливає з її назви: використовуються два криптографічні ключі публічний та приватний, інформацію, що шифрується публічним ключем можливо розшифрувати приватним і навпаки. Проте один і той же ключ не може використовуватися для шифрування/розшифрування. Таким чином веб-сервер зберігає в таємниці приватний ключ, а публічні віддає клієнтам (рис. 1.4). А процес обміну ключами та встановлення правил криптографії називається рукостисканням (*handshake*).

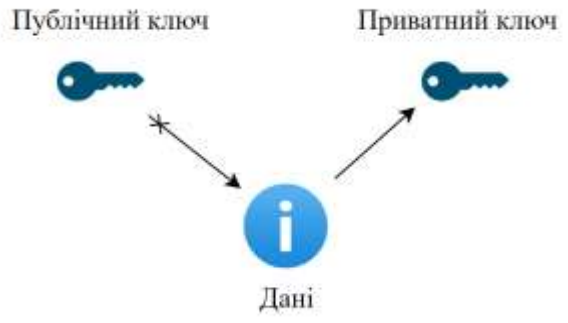

Рис.1.4. Процес шифрування даних публічним ключем

Протоколи *TLS/SSL* використовують асиметричну криптографію лише на самому початку сеансу зв'язку. Сервер та клієнт повинні домовитись про єдиний ключ сеансу, який вони обидва використовуватимуть для шифрування/дешифрування своїх пакетів. Шифрування за допомогою спільного ключа називається симетричною криптографією. Симетрична криптографія вимагає меншої обчислювальної потужності у порівнянні з асиметричною.

Оскільки шифрування та дешифрування за допомогою приватного та відкритого ключа вимагає значної обчислювальної потужності, вони використовуються лише під час процесу рукостискання для створення симетричного ключа сеансу. Після встановлення безпечного з'єднання ключ сеансу використовується для шифрування всіх переданих даних.

Процес рукостискання включає в себе етапи, що показані на рис. 1.5.

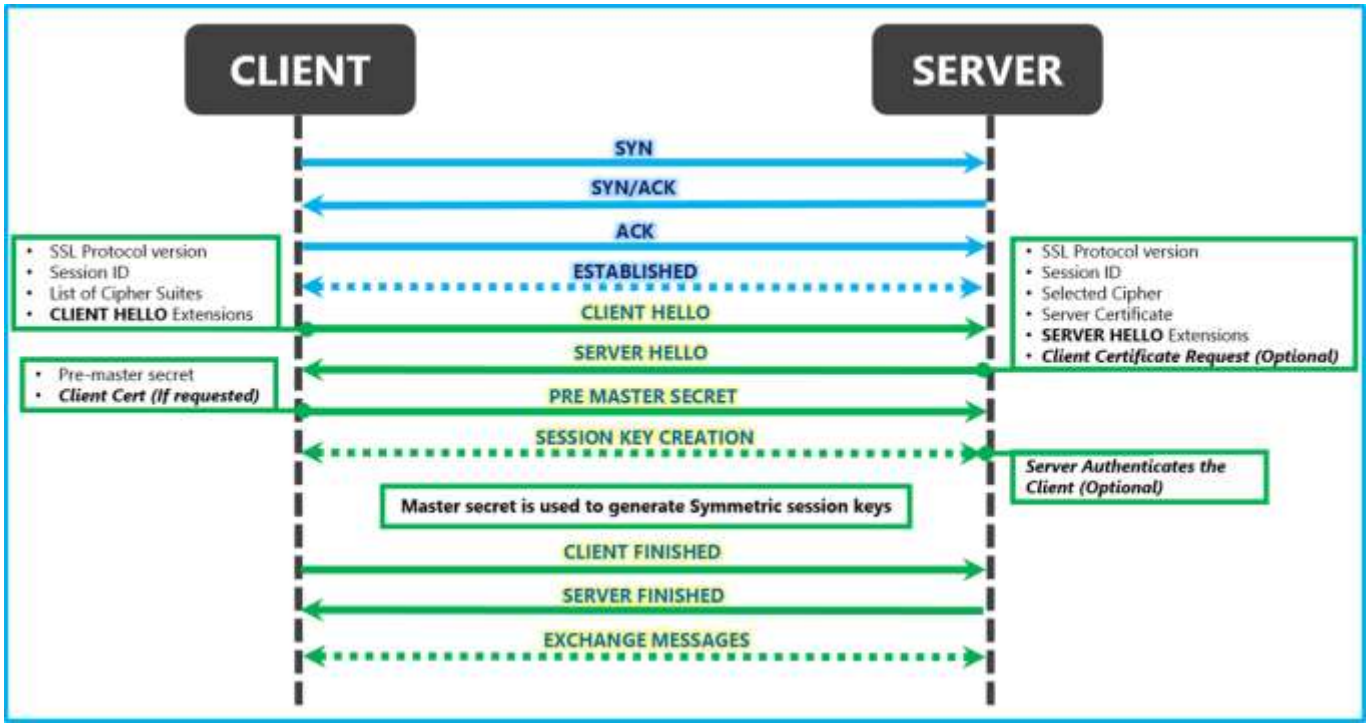

Рис.1.5. Візуалізація процесу *handshake*

1. Клієнт створює *TCP* з’єднання з веб-сервером та запитує процес рукостискання.

2. Веб-сервер відповідає списком наборів підтримуваних ним шифрів – алгоритмічних наборів інструментів для створення зашифрованих з’єднань.

3. Клієнт порівнює отриманий набор шифрів зі своїм власним списком підтримуваних шифрів та обирає один і повідомляє серверу, що вони обидва будуть його використовувати саме його.

4. Веб-сервер надсилає цифровий сертифікат. Отримавши сертифікат клієнт перевіряє його справжність.

5. Клієнт генерує ключ сеансу та використовує відкритий ключ веб-серверу для його шифрування.

6. Зашифрований ключ сеансу надсилається веб-серверу, який у свою чергу дешифрує його. Таким чином ключ сеансу використовується для шифрування даних протягом усієї сесії.

Ключ сеансу є одноразовим та діє лише у межах одного сеансу зв'язку. При перериванні комунікації з будь-яких причин, наприклад через проблеми з мережею, процес рукостискання повторюється та використовується вже інший ключ сеансу.

1.4. Механізми стиснення даних при обміні веб-ресурсами

Стиснення даних – це процедура перекодування, основна мета якої зменшення розміру даних. Процедура стиснення базується на усуненні надлишкової інформації, що міститься у вхідних даних.

Для стиснення даних використовуються спеціальні алгоритми. Процес стиснення базується на постійній заміні повторюваних фрагментів коротшими, або ж заміні значень даних, що трапляються часто – коротшими, а частіше – довшими (імовірнісне стиснення).

Для стиснення даних, що передаються згідно правил протоколу *HTTP* використовуються наступні параметри алгоритмів:

1. *gzip*;
2. *compress*;
3. *deflate*;
4. *identity*;
5. *br*.

Алгоритми стиснення даних повинні забезпечувати:

1. Зменшення об’єму початкової інформації.

2. Мінімально впливати на продуктивність системи.

Алгоритм *gzip* – це кодування Лемпеля-Зіва (*LZ77*) з 32-бітним *CRC*; compress – Лемпеля-Зіва-Велча; *deflate* – використовує структуру *zlib* з алгоритмом стиснення *deflate* (визначений у *RFC* 1951); *identity* – означає відсутність стискання; *br* – використання алгоритму *Brotli*. Для стиснення даних найчастіше використовується алгоритм *gzip*.

Алгоритм стиснення зазначається у *HTTP*-заголовку *Content-Encoding*.

Content-Encoding = <algorithm>

Клієнт може надсилати *HTTP*-заголовок *Accept-Encoding* для визначення підтримуваних ним алгоритмів стиснення.

Accept-Encoding = <list of supported algorithms>

Висновки за розділом

У першому розділі здійснено аналіз технологій та процесів розподілених веб-ресурсів: введено поняття веб-ресурсу, описані механізми обміну веб-ресурсами, розглянуті особливості шифрування та стиснення даних при обміні веб-ресурсами.

З’ясовано, що веб-ресурс – це узагальнене поняття одиниці корисної інформації, що може бути представлена у текстовому чи медіа форматі. Доступ до веб-ресурсів здійснюється за допомогою мережі та веб-ресурс ідентифікується у ній через унікальний ідентифікатор ресурсу – *URL*.

Особлива увага приділена механізмам обміну веб-ресурсами, оскільки для створення системи керування доступом до розподілених веб-ресурсів необхідно повністю розуміти усі механізми та правила обміну ними. Встановлено, що основний протокол для здійснення вищезазначених операцій – *HTTP*. Детально розглянуті деякі процеси, що використовуються протоколом *HTTP*. Вони будуть використані у подальшому під час розробки системи керування доступом до розподілених веб-ресурсів.

Додатково проаналізована тема шифрування та стиснення веб-трафіку перед їх передачею по мережі. Встановлено, що використання зазначених механізмів обґрунтовується збільшенням об’єму та розмірів веб-трафіку. Шифрування веб-трафіку обумовлюється створенням нових методів для несанкціонованого доступу до персональних даних користувачів під час обміну даними між клієнтами та веб-серверами. Стиснення даних обумовлюється зростаючими розмірами веб-ресурсів.

РОЗДІЛ 2
МЕХАНІЗМИ КЕРУВАННЯ ДОСТУПОМ ДО РОЗПОДІЛЕНИХ ВЕБ-РЕСУРСІВ

2.1. Мета керування розподіленими веб-ресурсами

Методологія обробки даних сучасних інформаційних систем заснована на принципі багаторівневості. Представлена на рис. 2.1 схема відображає рівні обробки інформації в інформаційній системі.

Рис.2.1 Схема передачі інформації в інформаційній системі [4]

Керування розподіленими веб-ресурсами передбачає аналіз інформації на рівні мережевого оточення операційної системи (ОС). При цьому сама система керування може використовувати техніки та технології з рівня прикладного програмного забезпечення (ПЗ), наприклад – СУБД.

Мета керування розподіленими веб-ресурсами, що розглядається у рамках дипломної роботи, – здійснення керуючих впливів на процес обміну веб-ресурсами для блокування визначених веб-ресурсів, виявлення інтернет-загроз у веб-трафіку та постійного моніторингу *HTTP*-з'єднань.

Розроблювана система керування доступом до розподілених веб-ресурсів є інформаційною системою. Зв'язки у системі будемо формувати з огляду на

архітектурну модель інформаційної системи (рис. 2.2) запропонованою автором Шелухіним [4].

Рис.2.2. Модель інформаційної системи

2.2. Дослідження керівних впливів

Доступ до веб-ресурсу – це можливість завантажити веб-ресурс за допомогою *URL*-посилання з мережі Інтернет. Доступ до веб-ресурсу може супроводжуватися додатковою необхідністю у авторизації та/або автентифікації.

Процес доступу до веб-ресурсу складається з кроків:

1. Завантаження клієнту та ввід бажаного *URL* веб-ресурсу користувачем.

2. Звернення клієнту до *DNS*-серверів або кешу ОС для трансляції доменного імені у *IP*-адресу (опціональний етап, описаний узагальнено).

3. Встановлення клієнтом *TCP*-з’єднання з веб-сервером (при цьому порт – 80 для нешифрованих та 443 для шифрованих з’єднань).

4. Клієнт створює *HTTP*-запит та надсилає його на веб-сервер.

5. Веб-сервер отримує *HTTP*-запит, аналізує його та формує *HTTP*-відповідь.

6. Веб-сервер надсилає *HTTP*-відповідь клієнту.

7. Клієнт отримує *HTTP*-відповідь, аналізує її та відображає отриману інформацію користувачу.

Відповідно до вищезазначеного процесу очевидним є те, що з метою мінімального втручання у роботу клієнта та веб-серверу доцільно здійснювати будь-який керівний вплив на етапі встановлення з'єднання між клієнтом і веб-сервером та під час передачі інформації між ними. Таким чином керівний вплив буде практично "прозорим" для обох сторін, що дозволить реалізувати додатковий функціонал у незалежний спосіб.

Проаналізувавши алгоритми роботи протоколу *HTTP* та описані вище процеси робимо висновки про перелік можливих керівних впливів на процес доступу до розподілених веб-ресурсів:

1. Створення та розривання *HTTP*-сесій.
2. Аналіз та/або редагування вмісту тіла *HTTP*-повідомлень.
3. Редагування формату даних *HTTP*-повідомлень.
4. Блокування та/або перенаправлення *HTTP*-трафіку.

Створення та розривання *HTTP*-сесій – це базовий керівний вплив на систему. Прийняття рішень про наявність чи відсутність зв'язку між вузлами мережі – це важлива можливість, що забезпечує керування процесом проходження трафіку у системи в будь-який період часу.

При найпростішому використанню даного керівного впливу, а саме для підтримки процесу передачі даних між вузлами без безпосереднього втручання у цей процес, реакція системи на вплив буде мінімальною.

Редагування вмісту тіла *HTTP*-повідомлень – складний керівний вплив, що безпосередньо втручається у інформацію системи. Результати даного впливу зазвичай чітко відображаються користувачам. Зазвичай рішення про застосування даного керівного впливу базується на комплексному аналізі веб-трафіку з певною метою.

Наприклад, аналіз веб-трафіку може здійснюватися з метою пошуку забороненої інформації у тілі повідомлення (небажані елементи на веб-сторінці або у веб-ресурсі), пошуку інтернет-загроз тощо. Зазвичай такий аналіз потребує значних апаратних та програмних ресурсів та може вносити значні затримки до всієї системи.

Замість даного впливу може застосовуватися керівний вплив розривання *HTTP*-сесій з метою спрощення процесів та зниження затримок.

Редагування форматів даних *HTTP*-повідомлень зазвичай використовується для спрощення процесів їх аналізу. Перед аналізом *HTTP*-повідомлення першочергово необхідно дізнатися, чи дійсно воно представлене у незашифрованому вигляді та чи не був використаний алгоритм стиснення на вмісті тіла повідомлення. Зашифроване повідомлення необхідно розшифрувати, а детальні відомості про цей процес будуть розглянуті у подальшому. Стиснене тіло повідомлення не несе у собі логічного вмісту оригінальної інформації, тому перед будь-яким аналізом веб-трафіку потрібно здійснити декомпресію даних попередньо визначивши потрібний алгоритм стиснення. Деталі цього процесу також розглядаються у подальшому.

Наприклад, *HTTP*-повідомлення може бути переформатоване з будь-якого формату у формат *Chunked Transfer Encoded*. Передача даних у даному форматі не вимагає знання довжини тіла повідомлення перед його передачею, а отже на самих даних у ході передачі можуть здійснюватися певні дії: стиснення частинами та інші перетворення. Таким чином знижуються затримки у системі при передачі по мережі, оскільки самі дані пересилаються постійно та невеликими пакетами.

Редагування форматів даних вимагає повного аналізу *HTTP*-повідомлення та розбиття його на складові частини. При зміні формату даних *HTTP*-заголовки та тіло повідомлення вимагають змін. Інакше кажучи – необхідна повноцінна реалізація тієї частини веб-серверу, що відповідає за аналіз та декомпозицію *HTTP*-запитів.

Блокування та/або перенаправлення *HTTP*-трафіку використовується у випадках спроб доступу до заборонених системою веб-ресурсів та при тимчасових змінах у мережевому місцезнаходженні потрібних веб-ресурсів.

Перший варіант керуючого впливу безпосередньо впливає на стан системи, оскільки користувач не зможе отримати запитаний веб-ресурс та інформація про це буде відображена у клієнті. Доцільним рішенням є надсилання *HTTP*-відповіді з кодом відповіді 403 *Forbidden*.

Другий варіант зазвичай відбувається максимально “прозоро” для кінцевого користувача, про що він може навіть не здогадуватись.

Згідно досліджених керівних впливів визначаємо, що керування доступом до розподілених веб-ресурсів може базуватися на великій кількості можливих тем та застосовуватися з різною метою. У дипломній роботі дослідимо наступні керівні впливи та теми:

1. Аналіз вмісту тіла *HTTP*-повідомлень на наявність певних інтернет-загроз.
2. Блокування *HTTP*-трафіку для визначених веб-ресурсів.
3. Редагування формату даних *HTTP*-повідомлень для їх підготовки до подальшого аналізу.
4. Створення та розривання *HTTP*-сесій для з'єднання клієнтів та веб-серверів, а також для реагування на результати аналізу вмісту тіла *HTTP*-повідомлень (веб-трафіку).

2.3. Аналіз поширених інтернет-загроз

Складність та масштаби кібератак постійно зростають. Прогрес створення шкідливого ПЗ пришвидшується, створюються його більш витончені види. Основне завдання шкідливого ПЗ – інфікування комп'ютерів користувачів. На сьогодні існують наступні типи інтернет-загроз:

1. Шкідлива програма (трояни, програми-вимагачі).
2. Фішинг.
3. Спам.
4. Прихований майнінг.
5. Крадіжка особистої інформації.
6. Бекдор.
7. Комп'ютерний вірус.
8. Експлойт.
9. Шпигунські програми.
10. Мережевий хробак.
11. Рекламне ПЗ.
12. Соціальна інженерія.

Частина із вищезазначених інтернет-загроз використовує методи, що розраховані не неуважність та необізнаність користувача (фішинг, крадіжка персональної інформації, соціальна інженерія), інша – програмні методи для виконання зловмисницьких дій.

У дипломній роботі досліджуються ті типи інтернет-загроз, які мають чіткі ознаки (сигнатури), та які можуть бути виявлені у ході аналізу веб-трафіку. До таких інтернет-загроз відносяться ті, що розповсюджуються у вигляді шкідливого ПЗ: шкідливі програми, прихований майнінг (зазвичай розповсюджуються у вигляді троянів), комп'ютерні віруси, шпигунські програми, мережеві хробаки та рекламне ПЗ.

Аналіз та виявлення інтернет-загроз зазвичай відбувається на кінцевих комп'ютерах користувачів, використовуючи їх апаратні ресурси. Для цього використовується персональне антивірусне ПЗ. Проте існує підхід, який дозволяє здійснювати уніфікований захист одразу декількох груп користувачів у межах локальної мережі. Він полягає у використанні спеціального ПЗ для аналізу трафіку на етапі його передачі по мережі. Таким чином при виявленні інтернет-загрози такий трафік може бути заблокований або замінений на нешкідливий. Системи, що використовують дані підходи: *SNORT* та *SURICATA*. При використанні даного підходу системи доцільно розміщувати на виділених серверах у межах локальної мережі для забезпечення ефективності їх роботи.

Отже, тема аналізу та виявлення інтернет-загроз у веб-трафіку тісно пов'язана з захистом комп'ютерних мереж, тому подальші дослідження потребують розгляду цієї теми.

2.4. Проблеми та мета захисту комп'ютерних мереж

Проблеми автоматизованого захисту комп'ютерних мереж від зовнішніх загроз постає однієї із ключових у зв'язку з збільшенням інтенсивності користування Інтернетом, розширенням каналів зв'язку, перенесенням інформаційного простору у онлайн.

З кожним днем усе більше користувачів з'являються у мережі Інтернет, активно розвиваються методи та технології доступу до Всесвітньої павутини (наприклад одним із останніх досягнень у цій галузі став супутниковий Інтернет *StarLink*). Відповідно виникають усе більші загрози використання Інтернету для самих користувачів.

Багато вчених вели дослідження на предмет виявлення несанкціонованих дій у комп'ютерних мережах протягом усієї історії їх розвитку.

Одна з перших робіт [5] охарактеризувала основні поняття несанкціонованих дій та надала рекомендації щодо їх вирішення. Проте надані рекомендації не створювали необхідного інструментарію для їх аналізу, а використані методи досліджень базувалися суто на теорії ймовірності.

Автор Шейнер [6] запропонував підхід виявлення вразливостей протоколів та програм базуючись на критеріях поведінки системи.

Деякі моделі засновані на аналізі неформальних параметрів – сигнатурних, а їх перше використання не забезпечувало отримання повноцінної оцінки завершеності та ефективності [7], хоча згодом модель [8] ввела можливості їхнього розрахунку.

У праці [9] автори притримуються думки, що будь-яке вторгнення базується на чітко окреслених станах та переходах. Проте ця праця не пропонує інструментарію для контролю вторгнень та не описує дії системи користувача.

Вибір методів аналізу та захисту комп'ютерних мереж від загроз залежить від сфери призначення. Менша сфера призначення спрощує завдання та полегшує процес аналізу, ширша сфера – навпаки. Саме від сфери призначення залежить підбір моделі поведінки мережевих об'єктів [10, 11].

До систем та методів виявлення загроз у комп'ютерних мережах також відносяться такі, що засновані на використанні нейронних мереж (НМ). Особливість НМ у тому, що вони починають працювати тільки після процесу навчання. Навчання засноване на принципі співвідношення вхідних та вихідних сигналів нейронів [12]. Через те, що НМ – це по суті чорний ящик, то неможливо аналітично прорахувати похибки аналізу. Основним недоліком використання НМ у даній сфері є те, що процес навчання НМ стає ефективним тільки після досить тривалого часу.

Узагальнено, основна мета системи виявлення загроз – це своєчасно аналізувати, виявляти та блокувати можливі загрози. Ранній етап виявлення можливої загрози забезпечує стабільність системи та зменшує негативні наслідки. Тому властивість виявлення загрози на ранніх стадіях відноситься до найважливіших у системах виявлення загроз. При низькому значенню цієї властивості загальна користь від такої системи падає у геометричній прогресії.

Підвищення складності програмного забезпечення (ПЗ) призводить до аналогічного ускладнення вірусного ПЗ. Вірусне ПЗ еволюціонує та його програмні елементи можливо знайти на чорних ринках ПЗ. Ці можливості завдають шкоди звичайним користувачам, корпораціям, установам, а іноді – економіці та стабільності цілих країн (приклад – вірус *Petya*). Програмні елементи вірусного ПЗ складаються з невеликих програмних блоків вбудованих до виконуваних файлів звичайних програм, з цілих систем взаємопов'язаних компонентів для виконання конкретних завдань: віруси-маскувальники, інсталятори тощо. Основне джерело вірусного ПЗ – це мережа Інтернет.

Способи та методи виявлення загроз у комп'ютерних мережах – різноманітні та багатогранні. Тим не менш, оцінка якості виконання їх прямих обов'язків не завжди достатня для загального оцінювання. Будь-який додатковий елемент у системі може призвести до непропорціонального росту її складності. Наряду з виконанням своїх прямих обов'язків такі системи також повинні:

1. Притримуватись принципу "прозорості" та мінімально втручатися у механізми самої мережі, оскільки втручання у роботу мережі веде до ускладнення її процесів та негативних наслідків.

2. Критичним пунктом є мінімальний вплив на затримку в мережі, оскільки питання продуктивності комп'ютерних мереж – це окрема галузь, на базі якої, наприклад, засновані критерії оцінки веб-додатків та веб-сайтів.

У зв'язку зі збільшенням загроз несанкціонованого доступу до даних користувачів та впровадженням нових стандартів по їх обробці та збереженню. Наприклад, європейський загальний регламент про захист даних (*GDPR*), що містить положення та вимоги щодо опрацювання особистої інформації користувачів та

вводить принцип “приватність за призначенням та за замовченням”, який означає, що персональні дані необхідно зберігати з використанням псевдонімів чи повної анонімізації та використовувати налаштування найвищого рівня приватності за замовчуванням, так щоб дані не були доступні публічно без очевидної згоди та не могли бути використані для ідентифікації суб'єкта без додаткової інформації.

Компанії змушені витрачати величезні кошти на:

1. Мережеву безпеку.

2. Підвищення кваліфікації працівникам *IT*-сектору у сфері комп’ютерної безпеки.

3. Захист та мінімізацію ризиків вірусних вторгнень.

4. Впровадження політик виявлення та запобігання вторгнень.

5. Резерви даних, правил та процесів їх відновлень.

6. Визначення політик та процедур безпеки.

Безпосередній несанкціонований доступ зазвичай отримується через ряд причин:

1. Слабкість в контрольованій системі з неефективними елементами управління.

2. Помилки в ПЗ та відсутність своєчасного виправлення цих помилок.

3. Необачність працівників системи.

Операційні системи піддаються несанкціонованому доступу при умові запуску на ній сторонніх додатків з шкідливим кодом. Такими додатками можуть бути різноманітні файли, як виконуючі, так і ні. У деяких випадках сторонні додатки виступають у вигляді окремих плагінів для інших програм, наприклад плагіни для *Microsoft Word, Microsoft Excel* тощо. Через загрозу вірусів розробник вищезазначених програм передав відповідальність за можливі несанкціоновані дії на користувача – просто надавши окрему опцію для вмикання плагінів, а за замовченням вони вимкнені. Тобто компанії з такими ресурсами як *Microsoft* не мають змоги повноцінно боротись з вірусами, що розповсюджуються через створене ними ж ПЗ, а основна реакція на це – лише ненадійна латка та перекидання відповідальності (рис. 2.3).

Рис.2.3. Попередження *Microsoft Excel* про можливі віруси у плагіні

Несанкціонований доступ може бути отриманий як до ресурсів самої ОС, так і до ресурсів окремих програм. Причини виникнення вищезазначених дій та зловживань описані у роботах авторів [13, 14], а необхідні умови для цього наступні:

1. Відкритий код ядра ОС (*opensource*).
2. Розповсюдженість та популярність ОС.
3. Велика кількість багів та помилок у ядрі ОС (як відомих, так і ні).

Отримання несанкціонованого доступу до ОС – це наслідок відсутності або ж слабкого захисту мережевого оточення комп'ютеру, на якій встановлена ця ОС. Тому є сенс не вирішувати наслідки, а знайти першопричину та боротися в першу чергу з нею.

Безпека будь-якої системи залежить від набору програмних та архітектурних рішень по запобіганню отримання доступу до її керуючого ядра та файлів. Часто керівні вузли мають параметри та налаштування, які для зручності та доступності зберігаються прямо в постійній пам'яті у вигляді файлів налаштувань. Хоча такий спосіб є загальноприйнятним з точки зору зручності користування будь-якою системою він привносить доволі небезпечні механізми, за допомогою яких частини системи, або ж уся система у цілому, може бути скомпрометована. Тому використання загальнодоступних файлів налаштувань є можливим рішенням при розробці системи, але використання подібного методу на доступній та працюючій системі з великою кількістю користувачів є неприйнятним. Потрібно пам'ятати, що отримати доступ до динамічної оперативної пам'яті набагато важче, ніж до статичної

файлової системи. Найкращий метод передачі стартових параметрів та налаштувань до програми з точки зору захисту конфігураційних налаштувань є командний рядок.

Стрімкий розвиток вірусів не дає часу компаніям, основний продукт яких – антивірусне ПЗ, на повний аналіз їх сигнатур. Результатом здорової конкуренції на ринку антивірусного ПЗ, конкретно в цій сфері, є нехтування якістю досліджень та пришвидшення оновлення баз сигнатур вірусів.

Методи знешкодження вірусів також починають бути неефективними. У результаті комерціалізації сфери шкідливого ПЗ з'явилися нові техніки, які забезпечують:

1. Приховування шкідливого коду серед звичайного.

2. Мутуючий код, який при запуску з звичайного перетворюється на шкідливий.

3. Перешкоджання видалення шляхом клонування та вимкнення деяких системних функцій.

4. Контроль цілісності вірусної системи.

Ці техніки негативно впливають на результати розпізнання та видалення вторгнень.

Споживання системних ресурсів також впливає на систему. Ефективність будь-якого захисту від інтернет-загроз залежить від моніторингу веб-трафіку у режимі реального часу, що вимагає активної роботи з системними механізмами та подіями для фільтрації потоків можливих загроз. Як показує практика, активне використання системних функцій призводить до сповільнення роботи програм. Тому більшість користувачів, з причини відсутності необхідних високошвидкісних елементів постійної пам'яті (*SSD*) та потужних *CPU*, вимикають будь-який захист ОС таким чином ставлячи себе під ризик зовнішніх загроз.

Також існує проблема несумісності різних антивірусних програм через конфлікти під час перехоплення та фільтрації системних подій. Загалом – це несуттєва проблема, проте певне антивірусне ПЗ добре захищає від одного типу загроз, а інше – від іншого, і як результат неможливо створити єдиний захисний екран в межах однієї антивірусної програми.

Постає необхідність у впровадженні нових методів боротьби з інтернет-загрозами у комп’ютерних мережах.

Для подальшого опрацювання цієї теми необхідно дослідити вже існуючі методи боротьби, визначитись з доцільним рівнем розташування таких методів у топології системи, проаналізувати механізми виявлення інтернет-загроз, з’ясувати впливи на продуктивність, обрати метод захисту.

2.5. Методи виявлення інтернет-загроз у комп’ютерних мережах

Сучасне вірусне ПЗ призначене для контролю та перегляду інформації у несанкціонований спосіб за допомогою застосування механізмів вторгнення до комп’ютерних систем та мереж.

Віруси поділяються на класи з загальними характеристиками [15, 16]:

1. Алгоритм роботи.
2. Середовище існування.
3. Негативні наслідки.

У свою чергу середовища існування:

1. Мережеві.
2. Файлові.
3. Завантажувальні.
4. Макровіруси.

Середовище існування обумовлює тип розповсюдження вірусу. Мережеві віруси зазвичай розповсюджуються через веб-сайти та поштові клієнти. Файлові вбудовуються у звичайні файли. Завантажувальні записуються у *boot*-секторі. Макровіруси передаються разом з текстовими документами та таблицями.

Нерідко виникає ситуація, коли віруси комбінуються. Комбінування вірусів відбувається з метою ускладнення їх виявлення та аналізу, а відповідно – це призводить до запізнення вчасного видалення такого вірусу з мережі та наслідків у вигляді вкрадених або зруйнованих даних користувачів. Прикладом такого вірусу є мережевий макровірус.

Алгоритми роботи вірусів характеризуються [17, 18]:

1. Стелс-алгоритмами.

2. Резидентністю.

3. Поліморфічністю та самошифруванням.

Резидентність віруса заснована на його можливості проникнення у оперативну пам’ять комп’ютера. Знаходячись у оперативній пам’яті, а відповідно маючи певні привілеї з боку ОС, такі віруси перехоплюють виклики системи до об’єктів та записуються до них.

Використання стелс-алгоритмів забезпечує приховування вірусів у системі. Стелс-алгоритми працюють у парі з іншими вірусами. Наприклад, вірус знаходиться у інфікованому ним файлі. При спробі доступу до цього файлу за допомогою викликів системних функцій ОС стелс-алгоритм приховає реальний інфікований файл, а у відповідь дасть неінфікований.

Поліморфічність та самошифрування використовуються з метою приховування вірусів, на кшталт стелс-алгоритмів. Проте механізми приховування при цьому відрізняються.

Зазвичай вищезазначені механізми використовують усі віруси для запобігання виявлення себе системою та антивірусами. Наприклад поліморфічні віруси не мають сигнатур та постійних ділянок коду. Два зразки одного і того ж поліморфічного вірусу можуть повністю не співпадати. Цей механізм реалізується за допомогою шифрування основного коду вірусу та модифікації частини-розшифровувача.

За негативними наслідками класифікуються [19]:

1. Безпечні віруси. Зменшують пам’ять диску, створюють графічні та аудіо артефакти.

2. Нешкідливі віруси. Зменшують пам’ять диску.

3. Небезпечні віруси. Створюють відмови у роботі системи.

4. Дуже небезпечні віруси. Руйнують дані, повністю знищують коректну роботу системи.

2.5.1. Сигнатурний аналіз

Сигнатурний аналіз вірусів призначений для виявлення вже відомих вірусів. Сигнатурою називають набір ознак, які характеризують певний об’єкт. Набір ознак допомагає коротко охарактеризувати великі об’єкти. Аналогічним прикладом слугують хеш-функції, що надають змогу унікально описати великий набір даних через невеликий. До слабких ознак (сигнатур) можна віднести розмір, дату створення, тип, адресу файлу.

Описані властивості сигнатур дозволяють чітко встановити ознаку вторгнення у мережу. Це вторгнення буде представляти собою набір однотипних та розташованих один за одним байтів. Найсильніша сигнатура вірусу при такому варіанті являтиме собою набір байтів, що описуватиме його ядро.

Звичайний розмір цільної сигнатури загрози може бути завеликим для її використання при аналізі, що не є доцільним з практичної точки зору. Тому для мінімізації їх розміру використовується метод фрагментації. Фрагментація сигнатур має на увазі використання переривчастих ознак з двома частинами:

1. Унікальними.
2. Загальними.

Загальні частини характеризують певний підвид вторгнень, такі частини можуть бути притаманні одразу декільком вірусам. Унікальні частини містять сигнатури унікальних вірусів, що не повторюються у інших.

На практиці існує також метод “половинчастих” сигнатур, що використовуються для виявлення поліморфних вірусів [14, 18, 19].

Основною перевагою сигнатурного аналізу є точність його роботи та чітке визначення типу вірусу. Ця перевага дає можливість вносити до бази даних (БД) як сигнатури, так і методи блокування вірусів.

Недоліками є:

1. Виявлення лише відомих та проаналізованих вірусів.
2. Необхідність постійного оновлення БД.
3. Для виокремлення сигнатур вірусів необхідно мати його робочий зразок.

2.5.2. Евристичний аналіз

Евристичний аналіз призначений для знаходження нових несанкціонованих дій у мережі [19, 20]. У його основі лежить дослідження виконуючих файлів, їх аналіз, а на виході отримання результату про наявність чи відсутність загроз.

Вищезазначений процес має наступні кроки:

1. Відбувається розпізнання та перетворення команд програми. Після цього проводиться їх умовне виконання та знаходяться можливі небезпечні дії з даними.

2. Інтерпретація. Використовується для знаходження поліморфних загроз у разі відкладеного виконання небезпечних дій. На цьому етапі відбувається реальний запуск програми та аналізується процес виконання команд. Виконання етапу рекомендується проводити у оточенні з обмеженим доступом (у так “пісочниці”).

3. Проведення прагматичного аналізу. Аналіз змісту команд та їх груп для виявлення алгоритму небезпечних дій та їх призначення.

4. Виявлення “дискретних” моделей вірусів [15] та розпізнання подій, що належать цим моделям. Існує статичний та динамічний семантичний аналіз. Перший передбачає аналіз коду програми на основі кінцевих автоматів без її запуску. Другий передбачає її запуск. Даний аналіз використовують більшість антивірусів, хоча і аналіз програм на основі кінцевих автоматів можливо легко обійти використавши інші функціональні методи для досягнення поставлених цілей. Тобто кінцевий автомат має враховувати усі функціональні та програмні методи, що можуть призвести його перехід з одного стану в інший, а це – неможливо.

5. Інтерпретація шкідливого коду емулятором. Емулятор (або ж “пісочниця”) створює окреме оточення від основної системи користувача, що унеможливлює доступ до неї. Тому таке оточення є ідеальним місцем для тестування поведінки вірусів, виконання їх програмних кодів та аналіз. Емуляція необхідна для точнішого аналізу програми, оскільки результати її виконання не завжди характеризуються початковими даними. Наприклад певні сигнатури вірусів з’являються лише у процесі їх роботи, після виконання деякої кількості команд.

До недоліків емуляції відносять:

– потребу у повному моделюванні фізичних та програмних вузлів ОС;

– зниження швидкодії через необхідність створення програмної моделі процесів ОС;

– глибина аналізу програми залежить від підтримуваних емулятором команд.

До загальних недоліків евристичних методів відносять:

1. Складність та сповільнення роботи системи.

2. Велика кількість помилкових спрацювань.

2.6. Інструментальні засоби для виявлення інтернет-загроз у комп'ютерних мережах

Існують різноманітні інструментальні засоби та методи, що дозволяють підвищити захист та надійність комп'ютерних систем та мереж. До них належать [21]:

1. Продукційні системи.

2. Статистичний метод.

3. Штучні нейронні мережі.

4. Мультиагентні системи.

5. Діагностика.

6. Імунні системи.

Деякі з вищезазначених методів можуть мати недоліки, що сповільнюють продуктивність усієї системи або жертвують точністю аналізу.

2.6.1. Продукційні системи

Продукційні системи – це системи, що використовують продукційну модель представлення знань [22]. Це найпоширеніша модель. Представлення знань у продукційних моделях співвідноситься із логічним виводом знань у логічних моделях. Оскільки процес уподібнюється до людського мислення, то розуміння та розробка процесів спрощується.

Система продукцій базується на простих тотожностях

$$IF\ A => B,$$

де *A* та *B* – це зв'язок "причина-наслідок".

Фактичне розуміння та інтерпретація продукції може відрізнятися в залежності від значення використаних операторів. Наприклад до правил продукції можливо віднести такі види операторів як *A* (структура або фрейм), *Y* – дія, що перетворює *A*. Тобто суто логічна інтерпретація накладає певні обмеження на оператори.

Узагальнено така модель має вид:

$$P = K, U, A => B, E$$

де *K* – клас ситуації;

U – умова активації;

A => *B* – суть продукції;

E – умова закінчення.

Спрощення продукційних моделей відбувається за допомогою порядку або пріоритету, що вводяться на усю множину продукцій. Порядок визначає умови використання наступної продукції (коли попередня не підходить). У пріоритетах першочергово використовується продукція з найвищим пріоритетом. Для боротьби з однаковими пріоритетами використовують повернення.

Компонентами продукційної моделі є:

1. Робоча пам'ять.

2. Механізм виведення.

3. База знань.

База знань використовує продукції та описує предметну область. Робоча пам'ять зберігає факти про поточний стан логічного висновку. Механізм виводу виконує підбір правил, що можуть виконати дані продукції.

Продукційні системи широко використовуються для сигнатурного аналізу [23]. Їх використання є дієвим лише при наявності вторгнень, що відбуваються за однаковим алгоритмом. Відомий сценарій порівнюється з діями в системі та робиться висновок про наявність несанкціонованих дій. Дані системи добре себе проявили у області виявлення та аналізу мережевих вторгнень (*AVZ*, *KasperskyLab* тощо).

За допомогою продукційних моделей можливий активний аналіз веб-трафіку під час роботи процесів та на відкритих портах. Їх основними перевагами є:

1. Фрагментарність. Опис окремих частин знань.

2. Інкрементність. Можливість створення нових правил незалежно від інших.

3. Прозорість. Вбудова моделі може відбуватись незалежно.

Основним недоліком є те, що загальний час виконання перевірок зростає з кількістю перевірочних правил, хоча цей недолік можливо врахувати та виправити в залежності від використання в конкретній системі.

2.6.2. Статистичний метод

Статистичний метод полягає у відслідковуванні появи характерних ознак для формування висновку про наявність вірусів. Метод формує імовірнісні висновки про наявність загрози, таким чином відслідковуючи коректні процеси системи та виявляючи відхилення в них. Наприклад, під час роботи системи формується таблиця частоти звернень до команд процесора. При різких змінах цієї частоти можна зробити висновок про наявність загрози. Таким чином відслідковується поведінка поліморфних вірусів.

Зазвичай метод базується на теорії ймовірності і описується рівнянням Байєса (2.1).

$$P(D_i|S_j) = P(D_i) * P(S_i|D_j)|(P(D_1) * P(S_j|D_1) + P(D_2) * P(S_j|D_2) + (...) \qquad (2.1)$$

де S_j – події;

D_i – діагноз;

$P(D_i|S_j)$ – ймовірність коректності i-го діагнозу;

$P(D_i)$ – ймовірність i-го діагнозу;

$P(S_j|D_i)$ – умовна ймовірність появи i-ї ознаки події j.

Процес знаходження імовірності появи вірусу полягає у наступному: формується список програм та вірусів, що позначається $P(D_i)$. Потім визначаються рейтинги вірусів – $P(S_k|D_i)$. Формується вибірка подій K, $S = \{S_1, S_2 \dots S_n\}$. Далі визначається ймовірність діагностування наявності вірусу D_i, для чого вираховуються

ймовірності $P(D_i|S_j)$ для S_j з множини S. Отримуємо рівняння (2.2) для знаходження ймовірності появи вірусу.

$$P(D_i|S) = P(D_i|S_1) * P(D_i|S_1) * P(D_i|S_2) * \dots * P(D_i|S_K) \quad (2.2)$$

Після вираховуємо ймовірність вторгнень та обираємо найбільшу.

2.6.3. Штучні нейронні системи

Штучна нейронна система (ШНС) – це аналог нейронної системи мозку, що призначена для формування вихідного результату на основі аналізу вхідних даних, який заснований на результатах попереднього навчального етапу.

Навчання – це особливість ШНС. Процес навчання формує взаємозв'язки між вхідними та вихідними сигналами ШНС. Виявляються додаткові залежності. Мережа може навчатися на вибірці з одних даних, а працювати – на іншій. Можливе часткове доповнення даних при їх відсутності. Процеси ШНС засновані на отриманні, обробці та узагальненні даних.

ШНС впроваджують для захисту комп'ютерних систем та мереж від вірусів та несанкціонованого доступу [23, 24].

Найголовнішим завданням при впровадженні ШНС є формування правильної вибірки даних для навчання, оскільки навчена ШНС не може змінити алгоритм своєї роботи. Вона лише отримує параметри на вході та формує результат на виході. Тому підбір правильної топології мережі є вирішальним фактором. При коректному навчанні ШНС та її впроваджені до захисту мережі можливо отримати задовільні результати швидкості та надійності.

До парадигм навчання відносяться: кероване, спонтанне, з підкріпленням. Для задач захисту мереж найчастіше використовувані парадигми – спонтанна та з підкріпленням.

Типи НМ, представлені на рис. 2.4.

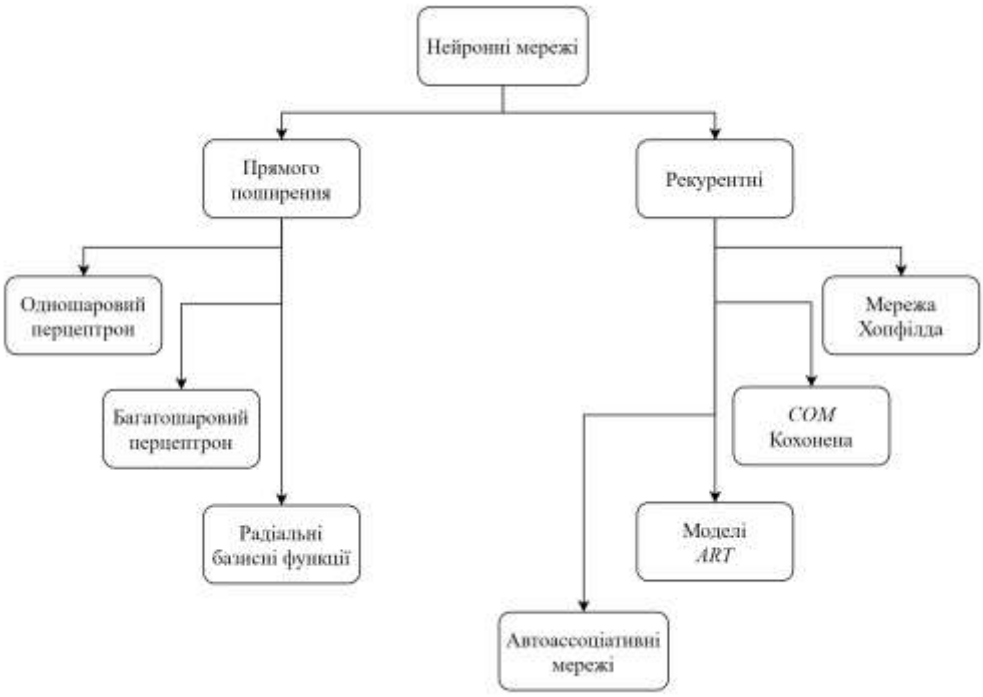

Рис.2.4. Типи нейронних мереж

Висновки за розділом

У другому розділі розглянуті питання механізмів керування доступом до розподілених веб-ресурсів: з'ясована мета керування розподіленими веб-ресурсами, досліджені можливі керівні впливи, проаналізовані сучасні інтернет-загрози, розкриті проблеми та мета захисту комп'ютерних мереж, розглянуті методи та інструментальні засоби виявлення інтернет-загроз у комп'ютерних мережах.

Основна мета керування розподіленими веб-ресурсами – здійснення керуючих впливів на процес обміну веб-ресурсами для блокування визначених веб-ресурсів, виявлення інтернет-загроз у веб-трафіку та постійного моніторингу *HTTP*-з'єднань.

Здійснено дослідження можливих керівних впливів: створення та розривання *HTTP*-сесій, аналіз та/або редагування вмісту тіла *HTTP*-повідомлень, редагування формату даних *HTTP*-повідомлень, блокування та/або перенаправлення *HTTP*-трафіку. Зроблений висновок про дослідження наступних керівних впливів: аналіз вмісту тіла *HTTP*-повідомлень на наявність певних інтернет-загроз, блокування

HTTP-трафіку для визначених веб-ресурсів, редагування формату даних *HTTP*-повідомлень для їх підготовки до подальшого аналізу.

Проведений аналіз сучасних інтернет-загроз, визначені конкретні для їх подальшого виявлення у веб-трафіку, зроблений висновок про зв'язок аналізу веб-трафіку на наявність веб-загроз та захист комп'ютерних мереж.

Розглянуті проблеми та мета захисту комп'ютерних мереж: кількість користувачів Інтернетом зростає, інтернет-загрози розвиваються. З'ясовано, що будь-яка система захисту повинна бути максимально "прозорою" та вносити мінімальне навантаження на ресурси.

Розглянуто методи та інструментальні засоби виявлення загроз у комп'ютерних мережах. Описані основні класи вірусів, їх характеристики, середовище існування. Розглянуті поняття сигнатурного та евристичного аналізу. Розглянуті інструментальні засоби по виявленню вірусів: продукційні системи (засновані на сигнатурному аналізі), статистичні методи, штучні нейронні системи та інші.

РОЗДІЛ 3
МЕТОДИ ТА ЗАСОБИ КЕРУВАННЯ ДОСТУПОМ ДО РОЗПОДІЛЕНИХ ВЕБ-РЕСУРСІВ

3.1. Технології проксі-серверів та веб-фільтрів

Проксі-сервер діє як шлюз між користувачем та Інтернетом. Це посередницький сервер, який відокремлює кінцевих користувачів від веб-ресурсів. Проксі-сервери забезпечують різний рівень функціональності, безпеки та конфіденційності.

Функціонал сучасних проксі-серверів набагато ширший, ніж виконання звичайних пересилань запитів. Проксі-сервери діють як брандмауер та веб-фільтр, забезпечують спільні мережеві підключення та кеш-дані для прискорення виконання запитів. Деякі проксі-сервери захищають користувачів та внутрішню мережу від мережевих вірусів, а також займаються питаннями конфіденційності..

Коли проксі-сервер отримує запит на веб-ресурс (наприклад, веб-сторінку), то він переглядає свій локальний кеш попередніх сторінок. Якщо сторінка знаходиться, то вона повертається користувачеві без пересилання запиту в Інтернет. Якщо сторінки немає в кеш-пам'яті, проксі-сервер, виступаючи клієнтом від імені користувача, використовує одну зі своїх власних *IP*-адрес для запиту сторінки (рис. 3.1).

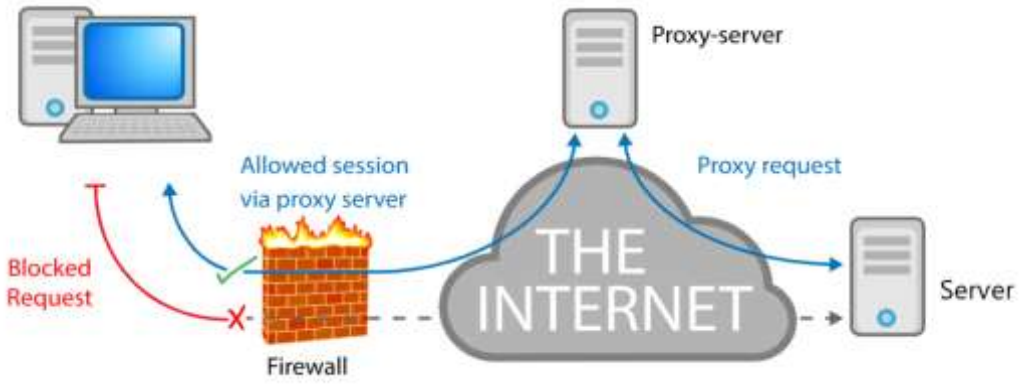

Рис. 3.1. Схема роботи проксі-сервера

Проксі-сервер працює абсолютно прозоро для користувача; всі Інтернет-запити та відповіді, що повертаються, безпосередньо стосуються адресованого Інтернет-сервера.

Користувачі можуть отримати доступ до веб-проксі через Інтернет або налаштувати веб-браузер для постійного використання проксі-сервера. Налаштування браузера включають автоматично виявлені та ручні параметри для проксі *HTTP, SSL, FTP* та *SOCKS*. Проксі-сервери можуть одночасно обслуговувати багатьох користувачів, або ж лише одного. Ці параметри відповідно називаються спільними та виділеними проксі.

Залежно від налаштування мережі та конфігурації існують наступні типи проксі-серверів:

Прямий проксі-сервери зазвичай використовуються внутрішніми мережами. Як тільки один із клієнтів надсилає запит на підключення до певного веб-сайту, він спочатку повинен пройти через переадресований проксі-сервер, який вирішує, чи дозволено клієнту звертатися до цього ресурсу. Якщо так, запит на підключення надходить на зовнішній сервер, який не бачить *IP*-адресу клієнта, а бачить лише запит на підключення, надісланий із переадресованого проксі-сервера. Прямий проксі-сервер забезпечує повний адміністративний контроль над підключеннями до локальної мережі. Він діє як брандмауер, що дозволяє адміністраторам обмежувати доступ до небажаних веб-ресурсів клієнтами внутрішньої мережі.

На відміну від прямого проксі-серверу, зворотній проксі-сервер працює на рівні вище веб-сайту (або веб-служби), приховуючи *IP*-адреси у внутрішній мережі від зовнішніх користувачів. Зворотний проксі-сервер вирішує, чи можуть веб-користувачі бачити вміст веб-сайту або використовувати веб-службу, чи ні. Зворотні проксі-сервера ускладнюють хакерам вторгнення на внутрішні ресурси. Крім того, зазвичай вони працюють як балансувачі навантаження для рівномірного розповсюдження даних між внутрішніми серверами, що запобігає перевантаженню тонами запитів на підключення. Зворотні проксі використовуються для:

1. Непрямого доступу.

2. Забезпечення балансування навантаження між серверами.

3. Потокової передачі контенту.

4. Заборони доступу до веб-сайтів.

Прозорі проксі зазвичай знаходяться біля шлюзу корпоративної мережі. Ці проксі централізують мережевий трафік. У корпоративній мережі проксі-сервер пов'язаний – або є частиною – шлюзового сервера, який відокремлює внутрішню мережу від зовнішніх, і брандмауер, який захищає мережу від вторгнень ззовні та дозволяє сканувати дані на безпечність перед доставкою клієнту в мережі. Ці проксі-сервери допомагають здійснювати моніторинг та адміністрування мережевого трафіку, оскільки комп'ютери в корпоративній мережі, як правило, є безпечними пристроями, які не потребують анонімності для звичайних звичайних завдань.

Анонімні проксі-сервери приховують *IP*-адресу клієнта. Вони можуть використовуватися для цілей конфіденційності та/або захисту від вторгнень.

Веб-фільтрація – це технологія, яка забороняє користувачам перегляд певних *URL*-адрес або веб-сайтів, блокуючи їх браузерам завантаження сторінок з цих веб-сайтів.

Веб-фільтри працюють за двома різними сценаріями. Вони можуть блокувати вміст сайтів відповідно до відомих списків класифікації сайтів. Або ж аналізувати вміст сайтів у режимі реального часу та вирішувати чи блокувати його, чи ні.

Типи веб-фільтрації:

1. Фільтр на основі списків дозволів.

2. Фільтри ключових слів та вмісту.

3. Окрім двох перелічених основних фільтрів, існують також варіанти налаштування: клієнтська фільтрація, серверна фільтрація, контроль шкідливих програм.

Дуже вичерпний огляд інструментів та технологій для фільтрації в Інтернеті описаний Мердоком та Андерсоном:

1. Фільтрація заголовків *TCP/IP*: За допомогою цього методу маршрутизатор може перевірити адресу *IP*-адреси та номер порту. Якщо місце призначення знаходиться у чорному списку, то з'єднання переривається.

2. Фільтрація вмісту *TCP/IP*. Це метод подібний до фільтрації заголовків *TCP/IP*, за винятком того, що маршрутизатор перевіряє вміст пакету на наявність будь-яких шаблонів або ключових слів, які можуть бути в чорному списку.

3. Фільтрування за допомогою проксі-серверів.

4. Гібридна фільтрація проксі-серверами, *TCP/IP* та *HTTP*.

5. Атаки *DoS*.

3.2. Порівняльний аналіз проксі-серверів

Apache та *Nginx* є двома основними використовуваними веб-серверами з підтримкою технологій проксі.

Apache (*Apache HTTP Server*) – високопродуктивний веб-сервер та зворотній проксі з відкритим сирцевим кодом, розроблений та підтримуваний *Apache Software Foundation*. *Apache* призначений для створення безпечного, надійного та ефективного комерційного веб-сервера відповідно до сучасних стандартів *HTTP*.

Apache залишається переважним вибором серед адміністраторів серверів завдяки своїй гнучкості, архітектурній простоті та підтримці різних платформ. Він підтримує майже усі популярні ОС. *Apache* у свій час став основою *WWW*, але з появою *NGINX* частка ринку *Apache* невпинно зменшується (рис. 3.2).

Рис. 3.2. Частота використання веб-серверів

Так, можна сказати, що *Nginx* не може конкурувати з багатофункціональним *Apache*, але його асинхронний підхід до обробки з'єднань та архітектура робить його розумним вибором у порівнянні з *Apache*.

Якщо взяти вибірку з рейтингу веб-сайтів в мережі Інтернет, то статистику найчастіше використовуваних веб-серверів можна побачити на рис. 3.3.

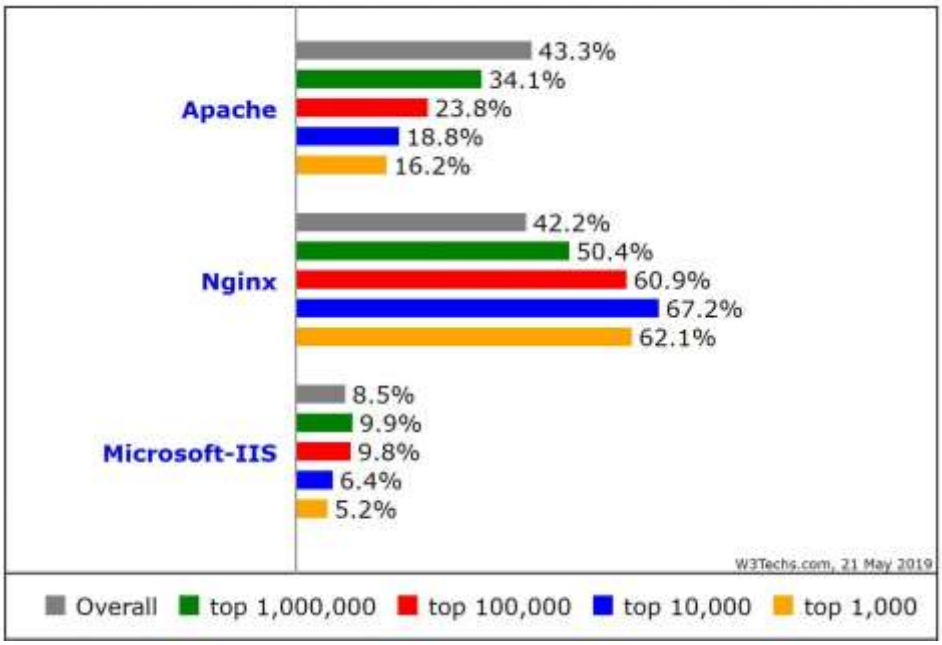

Рис. 3.3. Відсоткове порівняння популярності веб-серверів

Головні відмінності між *Apache* та *Nginx*:

1. *Apache* – це загальнодоступний *HTTP*-сервер, тоді як *Nginx* – це загальнодоступний високопродуктивний асинхронний веб-сервер та проксі-сервер.

2. Виправлення помилок, підтримка, обслуговування та розробка додатків на сервері *Apache HTTP* управляється та підтримується спільнотою користувачів з усього світу та координується *Apache Software Foundation*, тоді як *Nginx* управляється компанією з однойменною назвою.

3. Основна різниця між ними полягає в тому, як вони обробляють запити клієнтів. Незважаючи на те, що *Apache* пропонує різноманітність багатопроцесорних модулів для обробки запитів клієнтів та веб-трафіку, *Nginx* розроблений таким чином, щоб обробляти декілька запитів клієнта одночасно з використанням мінімальних апаратних ресурсів.

4. В *Apache* один потік пов’язаний лише з одним підключенням, тоді як один потік у *Nginx* може обробляти кілька підключень. Всі процеси розміщуються в циклі подій разом з іншими підключеннями і управляються асинхронно. Цей процес споживає менше пам'яті, збільшуючи тим самим продуктивність.

5. Сервер *Apache HTTP* має багатонитеву архітектуру, яка не масштабується. *Nginx* дотримується асинхронного *event-driven* підходу для обробки декількох запитів клієнтів. *Event-driven* архітектура розроблена для забезпечення кращої продуктивності навіть в умовах великого навантаження.

6. Сервер *Apache* обслуговує статичний вміст із використанням звичайних методів та обробляє динамічний вміст методами самого веб-сервера. *Nginx* не підтримує обробку динамічного вмісту та покладається на зовнішні процеси.

Найбільша різниця між *Apache* та *Nginx* полягає в базовій архітектурі способу обробки запитів. *Apache* обробляє запити за допомогою *MPM*-модулів або *Multi-Processing-Modules*, що відповідають за прив'язку до мережевих портів, прийняття запитів та відправлення дочірніх для обробки запитів. Найстаріший *MPM*, який бере свій початок аж до початку *Apache*, – це модуль *prefork*. У цьому режимі *Apache* породжує новий процес з одним потоком на кожен запит. Використання цього модулю означало вбудовування інтерпретатора *PHP* у кожен окремий процес, навіть для обслуговування файлів *CSS* та зображеннь, що є дуже неефективно. У наступні роки *Apache* розробив багатопотоковий *worker mpm*, а згодом - *event mpm*. Обидва вони полегшують багато проблем з продуктивністю *Apache*. Перехід на *php-fpm* дозволяє *Apache* залишатися конкурентним.

Nginx використовує асинхронну неблокуючу *event-driven* архітектуру, а *Apache* використовує процеси для кожного підключення (з *worker mpm* використовує потоки).

Nginx в ідеалі має один робочий процес на процесор/ядро. Відмінність робочих процесів *Nginx* полягає в тому, що кожен з них може обробляти сотні тисяч вхідних мережевих з'єднань. Немає необхідності створювати нові потоки або процеси для кожного з'єднання. Це є причиною того, чому основні *Content Delivery Networks*, такі

як *Cloudflare*, *MaxCDN*, *KeyCDN* – або веб-сайти, такі як *Netflix*, вважають, що *Nginx* має вирішальне значення для процесів представлення їх контенту.

3.3. Протоколи проксі-серверів

SOCKS – мережевий протокол, який призначений для реалізації доступу до сервісів, що знаходяться за брандмауерами.

Протокол не залежить від протоколів додатків та його можна використовувати для багатьох різних служб, таких як *telnet, ftp, finger*, *whois, gopher, WWW* тощо. Після встановлення з'єднання *TCP* з проксі-сервером він просто передає дані між клієнтом і сервером додатка, що вимагає мінімального споживання ресурсів. Оскільки *SOCKS* не повинен знати про протоколи обміну даних додатків, він також легко надає доступ додаткам з захищеними каналами зв'язку.

Клієнти за брандмауером, що потребують доступ до зовнішніх серверів, встановлюють з'єднання з *SOCKS* проксі-сервером. Такий проксі-сервер контролює права клієнта для доступу до зовнішніх ресурсів і передає запит до сервера. *SOCKS* може використовуватися і протилежним способом, дозволяючи зовнішнім клієнтам з'єднуватися з серверами за брандмауером.

На відміну від *HTTP* проксі-серверів, *SOCKS* передає всі дані від клієнта, нічого не додаючи від себе, тобто з точки зору кінцевого сервера, *SOCKS* проксі є звичайним клієнтом. *SOCKS* більш універсальний – не залежить від конкретних протоколів прикладного рівня (7-го рівня моделі *OSI*) і базується на стандарті *TCP/IP* – протоколі 4-го рівня. Зате *HTTP* проксі кешує дані і може ретельніше фільтрувати вміст переданих даних.

SOCKS 4 призначений для роботи через брандмауер без аутентифікації для додатків типу клієнт-сервер, що працюють за правилами протоколу *TCP*. По суті, *SOCKS*-сервер – це брандмауер, що підтримує протокол *SOCKS*.

Запити у протоколі *SOCKS* 4 визначаються чітко встановленими правилами. Запит клієнта при цьому містить:

1. Поле 1: номер версії, 1 байт (повинен бути 0x04 для *SOCKS* 4).

2. Поле 2: код команди, 1 байт:

– 0x01 = встановлення *TCP/IP* з'єднання;

– 0x02 = призначення *TCP/IP* порту (*binding*).

3. Поле 3: номер порту, 2 байти.

4. Поле 4: *IP*-адреса, 4 байти.

5. Поле 5: *ID* користувача, рядки змінної довжини, завершується *null*-байтом (0x00).

Відповідь сервера містить:

1. Поле 1: *null*-байт.

2. Поле 2: код відповіді, 1 байт:

– *0x5a* = запит наданий;

– *0x5b* = запит відхилений чи помилковий;

– *0x5c* = запит не вдався, бо не запущений *identd* (або не доступний з сервера);

– *0x5d* = запит не вдався, оскільки клієнтський *identd* не може підтвердити ідентифікатор користувача в запиті.

3. Поле 3: 2 довільних байти, ігноруємо.

4. Поле 4: 4 довільних байти, ігноруємо.

Протокол *SOCKS* 5 розширює *SOCKS* 4, додаючи до неї підтримку *UDP*, забезпечення механізмів автентифікації, розширює методи адресації додаючи підтримку доменних імен та адрес *IPv6*. Встановлення зв'язку складається з:

1. Клієнт підключається та посилає запит з переліком підтримуваних методів автентифікації.

2. Сервер вибирає один з них.

3. В залежності від обраного методу, між клієнтом і сервером може пройти деяка кількість повідомлень.

4. Клієнт посилає запит на з'єднання.

5. Сервер відповідає.

Методи автентифікації:

– *0x00* – без автентифікації;

– *0x01 – GSSAPI*;

– *0x02* – ім’я користувача та пароль;

– *0x03-0x7F* – зарезервовано *IANA*;

– *0x80-0xFE* – зарезервовано для методів приватного використання.

Запит від клієнта:

1. Поле 1: номер версії (повинен бути *0x05* для цієї версії).

2. Поле 2: кількість методів автентифікації, 1 байт.

3. Поле 3: номери методу автентифікації.

Сервер повідомляє:

1. Поле 1: Версія *SOCKS*, 1 байт (*0x05*).

2. Поле 2: обраний метод автентифікації, 1 байт, або *0xFF*, якщо не було запропоновано прийнятного методу.

3. Подальша ідентифікація залежить від обраного методу.

HTTP-проксі пересилають запити *HTTP*. Формат запиту до *HTTP*-проксі від клієнта є абсолютно однаковим зі звичайним *HTTP*-запитом до веб-серверу, за винятком того, що замість *URI* в заголовку повідомлення передається повноцінний *URL*. Наприклад:

GET https://www.priklad.org/Proxy_server HTTP/1.1

Proxy-Authorization: Basic encoded-credentials

Запит такого формату надсилається на проксі-сервер, проксі робить вказаний запит і повертає відповідь від безпосереднього веб-сервера.

Узагальнено підхід з встановленням з’єднання називається тунелювання *HTTP* (*HTTP tunneling*). Його використання може забезпечити обхід брандмауерів та використання заборонених протоколів у мережі.

Загальною формою встановлення з’єднання, додатково до вищеописаної, з *HTTP*-проксі вважається метод *HTTP CONNECT*. Механізм полягає у запиті клієнтом проксі-сервера щодо перенаправлення *TCP*-з’єднання до бажаного хосту. Проксі-сервер у свою чергу встановлює з’єднання на користь клієнта з бажаним хостом. Після цього процесу проксі-сервер лише пересилає потік даних по *TCP*-каналу між клієнтом та сервером. Протокол *HTTP* використовується лише на початку встановлення з’єднання для передачі інформації про віддалений хост та веб-ресурс, а

далі проксі-сервер лише підтримує *TCP*-з'єднання. Додатково до керування з'єднаннями *HTTP*-проксі може керувати доступами, блокувати веб-трафік, контролювати шифрування та інші аспекти при обміні даними. Приклад команди для встановлення з'єднання:

CONNECT priklad.com:443 HTTP/1.1

Proxy-Authorization: Basic encoded-credentials

Якщо з'єднання було успішним та проксі-сервер встановив зв'язок з віддаленим хостом, то проксі повертає:

HTTP/1.1 200 OK

Відповідно до специфіки поставлених завдань дипломної роботи доцільно буде використати технологію *HTTP*-проксі.

3.4. Вплив технологій шифрування та стиснення даних на процес аналізу веб-трафіку

Дослідження керівних впливів на доступ до розподілених веб-ресурсів показало, що аналіз мережевого трафіку є необхідним процесом для використання більшості із них.

Згідно правил протоколу *HTTP* дані *HTTP*-повідомлень можуть передаватися у шифрованому або стисненому вигляді. Ці властивості є корисними для кінцевого користувача. Проте з точки зору аналізу веб-трафіку вони привносять ускладнення, оскільки усі дані перед процесом аналізу повинні пройти додаткові етапи обробки.

Розглянемо вплив шифрування даних. Шифрування даних забезпечує захист та приховує їх зміст від третіх осіб.

Створювана система керування доступом до розподілених веб-ресурсів є свого роду *man-in-the-middle*. Механізми шифрування даних першочергово призначені для забезпечення захисту від атак типу *man-in-the-middle* (рис. 3.4).

Рис.3.4. Система керування доступом до розподілених веб-ресурсів з точки зору *MITM*

Тому при стандартному підході до встановлення *TCP*-з'єднань буде неможливо продивлятись та аналізувати веб-трафік, оскільки він буде зашифрованим. Процес рукостискання відбудеться між клієнтом та веб-сервером, а сама система не буде приймати участь у ньому. Єдиним рішенням є використання окремих з'єднань для клієнта та системи та для системи та веб-серверу. Таким чином система керування доступом отримає можливість розшифровувати дані. Недоліком такого підходу є те, що клієнт не зможе автентифікувати систему керування доступом за допомогою *SSL*-сертифікату, що мав би надати веб-сервер. Тобто з точки зору клієнта з'єднання залишиться незахищеним, про що клієнт обов'язково повідомить користувача.

Розглянемо вплив стиснення даних. Стиснення даних пришвидшує обмін даними, особливо у випадку з вузьким каналом для передачі даних.

Якщо до передачі дані були стиснуті, то для аналізу веб-трафіку першочергово необхідно здійснити процедуру декомпресії даних (рис. 3.5).

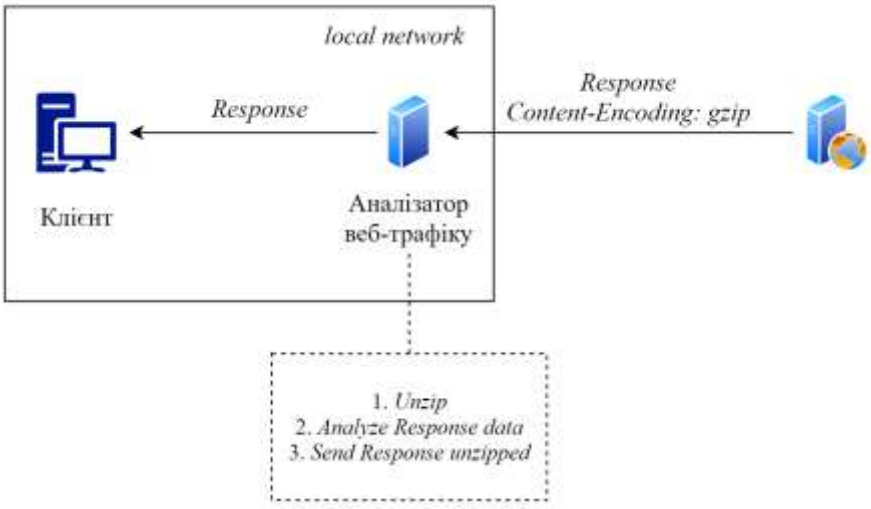

Рис.3.5. Передача даних у стисненому форматі та його вплив на аналіз веб-трафіку

Декомпресія даних займає додатковий час та, відповідно, створює затримки у отриманні даних клієнтом. Даний процес можливо оптимізувати, якщо у ході аналізу веб-трафіку тіло повідомлення буде перетворено у формат *Chunked Transfer Encoding* та частини даних після їх аналізу одразу відправлятимуться клієнту. *HTTP*-заголовок *Content-Encoding* при цьому необхідно прибрати з *HTTP*-відповіді. Реалізація цієї оптимізації буде доцільною тільки у випадку знаходження системи керування доступом до розподілених веб-ресурсів та клієнта у межах однієї локальної мережі. У іншому випадку таке рішення може призвести до зворотного ефекту та сповільнити процес обміну даними між усіма вузлами.

3.5. Статистична модель аналізу мережевого трафіку

Методи та моделі, що використовуються при виявленні та попередженні мережевих вторгнень (МВ) базуються на хостовому та мережевому аналізі сигнатурних та статистичних даних мережевого трафіку з послідуючим записом результатів аналізу до консолі або в журнал з інформацією про:

1. Час виявлення вторгнення.
2. Назву та тип вторгнення.

Засоби виявлення мережевих вторгнень (ЗВМВ) також можуть фіксувати низькорівневі дані про:

1. Номери мережевих пакетів.
2. Джерела вторгнення у вигляді *IP*-адрес та портів.

Сигнатурний аналіз та контроль профілів при виявленні мережевих вторгнень включає в себе аналіз заздалегідь відомих послідовностей як в самих аналізованих даних, так і в послідовності дій. Сучасні методики виявлення мережевих вторгнень досить різноманітні та не об’єднані у певні критерії для чіткої класифікації ефективності їх використання. Таким критерієм може слугувати повнота охвату усіх аналізованих параметрів необхідних для точного та найбільш вірогідного виявлення вторгнень з мінімальними похибками.

Недоліками статистичних моделей аналізу є велика кількість помилкових спрацювань та помилок другого роду, а сигнатурних – неможливість самостійного виявлення нових типів вторгнень та необхідність у постійному оновленні бази даних сигнатур.

При виявленні та попередженні мережевих вторгнень вже використовуються методики, що мають змогу саме попереджати МВ та виконують лише такі дії як блокування прийому/передачі тих мережевих пакетів, які ідентифікуються як загрозливі.

На сьогодні неможливе використання формалізованих методик та моделей для виявлення та попередження МВ через їх відсутність, що у свою чергу не дає точної змоги виявити МВ як під час роботи, так і під час розробки ЗВМВ. У відповідності до математичної моделі процесів зв'язку вузлів в мережі при виявленні та попередженні МВ розглядається наступна формалізація:

1. Мережевий трафік розглядається як сукупність повідомлень *S*, що позначаються як $S_{k,1}^{n_{U_{k,1}}}$, де $n_{U_{k,1}}$- номер повідомлення від U_k – останнього по порядку джерела повідомлення до U_1 – першому по порядку джерелу повідомлень, де k – кількість вузлів в інформаційній системі.

2. Ймовірність прийому одного повідомлення P після прийому/передачі попереднього повідомлення позначається як $P_{k,1}^{n_{U_{k,1}}}$- ймовірність прийому повідомлення $S_{k,1}^{n_{U_{k,1}}}$ з порядковим номером $n_{U_{k,1}}$ після прийому повідомлення $S_{k,1}^{n-1_{U_{k,1}}}$ від k-того вузлу до першого. Матриця таких перехідних ймовірностей виглядає як показано на рис. 3.6.

3. Стан вузлів інформаційної системи позначається як *Q*, Q^0 – нульовий стан та $Q_{k,1}^{n_{U_{k,1}}}$ – стан з відповідними індексами. Матриця станів виглядає як показано на рис. 3.7.

$$P = \begin{vmatrix} P_{2,1}^{1}, P_{2,1}^{2}, P_{2,1}^{3}, \ldots, P_{2,1}^{n_{U_{2,1}}} \\ P_{1,2}^{1}, P_{1,2}^{2}, P_{1,2}^{3}, \ldots, P_{2,1}^{n_{U_{1,2}}} \\ \text{----------} \\ P_{1,k}^{1}, P_{1,k}^{2}, P_{1,k}^{3}, \ldots, P_{1,k}^{n_{U_{1,k}}} \\ P_{k,1}^{1}, P_{k,1}^{2}, P_{k,1}^{3}, \ldots, P_{k,1}^{n_{U_{k,1}}} \end{vmatrix}$$

Рис.3.6. Матриця перехідних ймовірностей мережевих повідомлень

$$Q = \begin{vmatrix} Q_{2,1}^{0}, Q_{2,1}^{1}, P_{2,1}^{2}, \ldots, Q_{2,1}^{n_{U_{2,1}}} \\ Q_{1,2}^{0}, Q_{1,2}^{1}, Q_{1,2}^{2}, \ldots, Q_{2,1}^{n_{U_{1,2}}} \\ \text{----------} \\ Q_{1,k}^{0}, Q_{1,k}^{1}, Q_{1,k}^{2}, \ldots, Q_{1,k}^{n_{U_{1,k}}} \\ Q_{k,1}^{0}, Q_{k,1}^{1}, Q_{k,1}^{2}, \ldots, Q_{k,1}^{n_{U_{k,1}}} \end{vmatrix}$$

Рис.3.7. Матриця станів вузлів інформаційної системи

4. Статистичні показники *T*, сукупність яких позначається як $T=\{T1, T2, \ldots T_h\}$, де *h* – кількість статистичних показників. До таких показників при розгляді мережевого трафіку відносяться:

– кількість вхідних *IP*-пакетів за одиницю часу;

– кількість вихідних *IP*-пакетів за одиницю часу;

– кількість вхідних *TCP*-пакетів за одиницю часу;

– кількість вихідних *TCP*-пакетів за одиницю часу;

– кількість вхідних *UDP*-пакетів за одиницю часу;

– кількість вихідних *UDP*-пакетів за одиницю часу;

– час отримання пакетів;

– час відправлення пакетів;

– тривалість мережевого сеансу зв’язку;

– ймовірності *P*;

– стани *Q*.

5. Сигнатури позначаються як сукупність $M=\{M1, M2, \ldots M_g\}$, де g – кількість сигнатур вторгнень. Наприклад M_1 розглядається як наступні характеристики та параметри:

– поле “адреса відправника”;

– поле “адреса отримувача”;

– поле “тип”;

– поле “дані”;

– поле “*CRC*”;

– безпосередньо дані пакетів;

– час отримання пакетів;

– час відправлення пакетів;

– тривалість мережевого сеансу зв’язку.

M_2 розглядається як:

– поле “адреса відправника”;

– поле “адреса отримувача”;

– безпосередньо дані пакетів тощо.

При такій формалізації можливо визначити матриці сукупностей сигналів та статистичних показників. Матриці сукупності сигнатур *M* виглядає як показано на рис. 3.8.

	M_1	M_2	...	M_g
M_1	×	1	...	0
M_2	×	×	...	0
...	...	...	×	0
M_n	×	×	×	×

	M_1	M_2	...	M_g
M_1	×	1	...	1
M_2	×	×	...	1
...	...	...	×	1
M_g	×	×	×	×

Рис.3.8. Матриця сукупності сигнатур *M*

Елементи матриці сукупності сигнатур *M* обчислюються за правилами, що показані на рис. 3.9.

$$\begin{cases} M_{i,j} = 1, \text{ если } M \in \{M_1 \cup M_2, M_1 \cup M_3, M_1 \cup M_g, ..., M_1 \cup M_2 \cup ... \cup M_g\} \\ M_{i,j} = 0, \text{ если } M \notin \{M_1 \cup M_2, M_1 \cup M_3, M_1 \cup M_g, ..., M_1 \cup M_2 \cup ... \cup M_g\} \end{cases}.$$

Рис.3.9. Правила обчислення матриці сукупності сигнатур *M*

Матриці статистичних показників *T* виглядають як показано на рис. 3.10.

	T_1	T_2	...	T_h
T_1	×	1	...	0
T_2	×	×	...	0
...	...	...	×	0
T_h	×	×	×	×

	T_1	T_2	...	T_h
T_1	×	1	...	1
T_2	×	×	...	1
...	...	...	×	1
T_h	×	×	×	×

Рис.3.10. Матриці статистичних показників *T*

Елементи матриці статистичних показників *T* обчислюються як показано на рис. 3.11.

$$\begin{cases} T_{i,j} = 1, \text{ если } T \in \{T_1 \cup T_2, T_1 \cup T_3, T_1 \cup T_h, \ldots, T_1 \cup T_2 \cup \ldots \cup T_h\} \\ T_{i,j} = 0, \text{ если } T \notin \{T_1 \cup T_2, T_1 \cup T_3, T_1 \cup T_h, \ldots, T_1 \cup T_2 \cup \ldots \cup T_h\} \end{cases}.$$

Рис.3.11. Правила обчислення матриці статистичних показників T

Веб-трафік аналізується згідно вищеописаної математичної моделі. При виявленні вторгнення здійснюється блокування потрібного сеансу зв'язку між вузлами інформаційної системи або ж відбувається заміна даних в мережевих пакетах на такі, що не містять сигнатур вторгнення. Алгоритм дій для аналізу трафіку з метою виявлення загроз представлений на рис. 3.12.

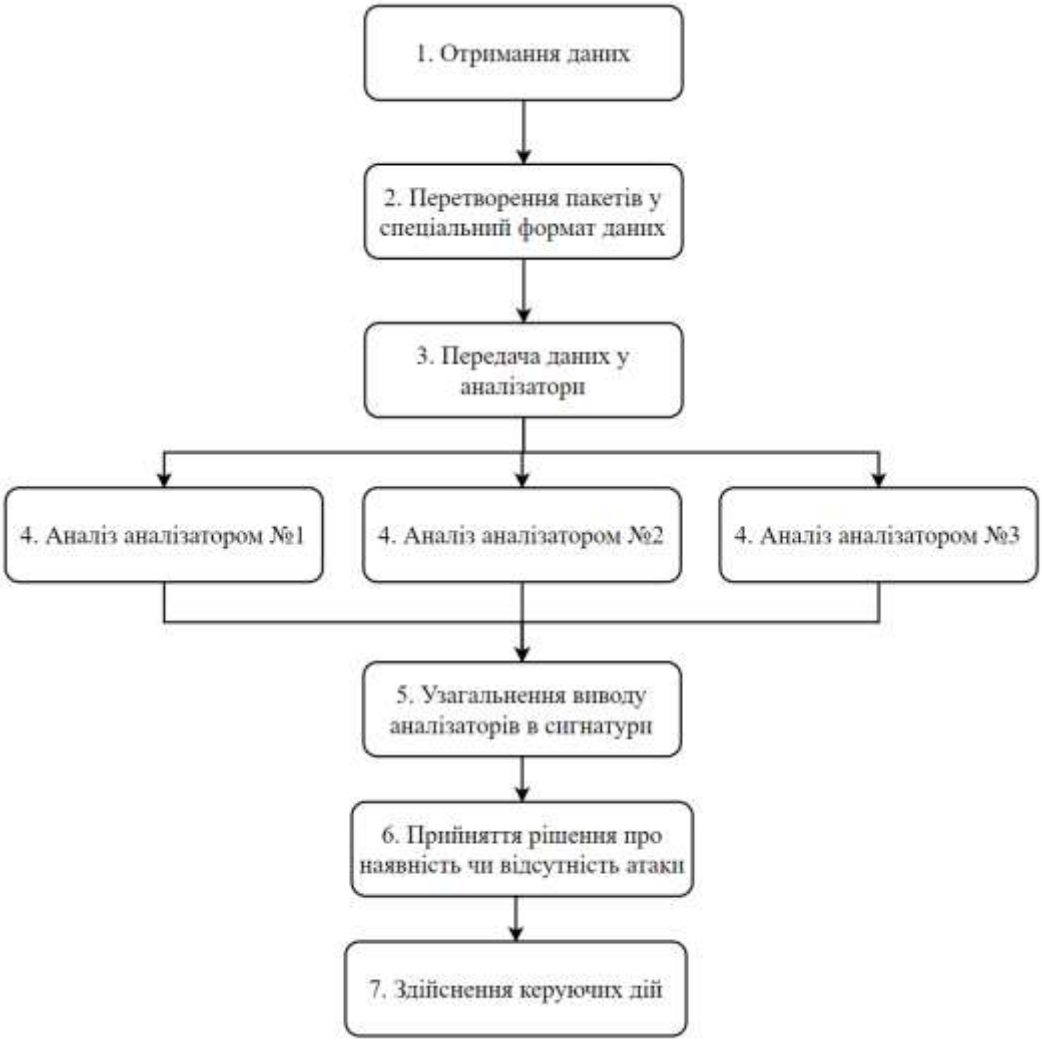

Рис.3.12. Алгоритм виявлення інтернет-загроз

Висновки за розділом

Третій розділ розкрив тему методів та засобів керування доступом до розподілених веб-ресурсів: розглянуті технології та сфери застосування проксі-серверів та веб-фільтрів, проведений порівняльний аналіз проксі-серверів, детально розглянуті протоколи проксі-серверів, досліджений вплив технологій шифрування та стиснення даних на процеси аналізу веб-трафіку, описана ститистична модель аналізу мережевого трафіку.

У результаті встановлено функціональне призначення проксі-серверів, їх типи: прямі та зворотні, зроблений опис відмінностей типів проксі, окрема увага приділена поняттю “прозорий проксі”. Розкрите питання мережевих фільтрів, а саме їх призначення та функціоналу.

Здійснено аналіз сучасних проксі-серверів. Для прикладу було взяті веб-проксі *Apache* та *NGINX*. З’ясовано, що дані програмні продукти широко використовуються при реалізації технологій доступу до веб-серверів. Здійснено їх порівняння, встановлені основні технології роботи, зроблено висновок про використання їх технологій у ході дипломної роботи.

Здійснений опис сучасних протоколів проксі: *Socks* 4, *Socks* 5, *HTTP*-proxy. Розписані основні механізми встановлення з’єднань та основне призначення. Прийнято та обґрунтовано рішення щодо використання протоколу *HTTP-proxy* у розроблюваній системі керування доступом до розподілених веб-ресурсів.

Досліджений та розкритий вплив технологій шифрування та стиснення даних показав, що розроблені механізми для покращення досвіду користувача ускладнюють процеси аналізу веб-трафіку.

Розглянута статистична модель аналізу мережевого трафіку у режимі реального часу методом сигнатурного аналізу. На основі моделі був створений алгоритм аналізу веб-трафіку, що буде використаний при розробці системи керування доступом до розподілених веб-ресурсів.

РОЗДІЛ 4
РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ДОСТУПОМ ДО РОЗПОДІЛЕНИХ ВЕБ-РЕСУРСІВ

4.1. Постановка вимог

При розробці системи керування доступом до розподілених веб-ресурсів необхідно врахувати та вирішити наступні задачі:

1. Передача даних.

2. Аналіз та обробка даних.

3. Зберігання даних.

Відповідно до вищеописаних задач розроблювана система повинна реалізовувати методи та механізми, що:

1. Реалізують механізми передачі даних.

2. Реалізують методи аналізу даних з метою подальшого прийняття керуючих рішень.

3. Зберігають результати перевірок та ведуть журнал дій.

Сформуємо функціональні вимоги. Необхідно спроектувати та реалізувати систему керування доступом до розподілених веб-ресурсів, що виконуватиме наступні завдання:

1. Аналіз вхідного мережевого трафіку на наявність інтернет-загроз (вірусів, хробаків, троянів тощо) методом сигнатурного аналізу згідно алгоритму запропонованого у розділі 3.5.

2. Блокування доступу до веб-ресурсів з можливістю ручного налаштування списку таких веб-ресурсів.

3. Здійснювати постійний моніторинг веб-трафіку та вести записи по основним мережевим характеристикам кожного вузла у окремий журнал (кількість вузлів, кількість вхідного/вихідного мережевого трафіку по кожному вузлу, виявлені інтернет-загрози).

Процес розробки системи вимагає:

1. Формування специфікації ПЗ.
2. Проектування ПЗ.
3. Програмування та тестування ПЗ.
4. Розгортання ПЗ.

4.2. Обґрунтування вибору технологій

Вибір технологій для розробки ПЗ є важливим етапом, що визначає подальшу можливість реалізації функціональних можливостей, безпосередньо впливає на швидкість його розробки, обумовлює потрібну кількість робочої сили, спрощує або ускладнює усі процеси розробки.

Наприклад, вибір мови програмування обумовлюється наступними факторами:

1. Підтримка *IDE*.
2. Кінцева вартість розробки ПЗ.
3. Можливість використання *best practices*.
4. Функціонал стандартної бібліотеки.
5. Тип мови (компільована/інтерпретована).

Область, до якої відноситься система керування розподіленими веб-ресурсами, є областю створення мережевих додатків, системного програмування та обробки інформації. Відповідно до цього раціональним рішенням є розгляд використання мови *C++* з надбудовою *Qt Framework*.

Оскільки мова *C++* – це низкорівнева компільована кросплатформена мова, то вона задовольняє вимоги для вирішення поставлених задач. Недоліками її використання стає факт відсутності у стандартній бібліотеці інструментарію по розробці мережевих додатків, а саме – сокетів. Для вирішення цього недоліку використаємо *Qt Framework*, а саме – його модуль *Qt Network*.

Qt Network надає набір *API* для створення додатків, що використовують *TCP/IP*. Операції *HTTP*-запитів та *HTTP*-відповідей, робота з файлами *cookie* та надсилання

даних обробляються різними класами *C++*. Для виконання поставлених завдань будуть використані наступні класи:

1. *QTcpSocket, QSslSocket* – класи, що використовуються для встановлення *TCP* та захищених *SSL* з'єднань.

2. *QHostAddress, QHostInfo* – класи з інформацією про *IP*-адреси та адреси хостів.

3. *QSslCertificate* – клас для керування сертифікатами *SSL*.

Також *Qt Framework* містить модуль *QtSQL*, що підтримує бази даних *SQL*. Робота з базами даних поділяється на три шари: рівень драйверу БД (може бути *SQLite3, PostgreSQL, Microsoft SQL Server* тощо), *SQL API*, додатковий інтерфейс для використання *SQL API*.

У зв'язку з характером розроблюваної системи необхідно використати локальну високошвидкісну БД з високою стійкістю до збоїв.

SQLite3 – це реляційна СУБД, що написана на мові *C*. Дана СУБД повністю відповідає вище заявленим вимогам.

Особливості *SQLite3*:

1. Повна підтримка концепції *ACID*.

2. Реалізація стандарту *SQL*-92.

3. Файл БД зберігається кросплатформеним файлом з підтримкою максимального розміру до 281 ТБ.

4. Повністю вбудовується у додаток та не має зовнішніх залежностей.

Для зберігання файлів конфігурації буде використано формат *JSON*. Для зчитування даного формату *Qt Framework* пропонує класи у модулі *QtCore*: *QJsonDocument, QJsonObject, QJsonArray, QJsonValue*.

4.3. Проектування системи

Створення системи керування доступом до розподілених веб-ресурсів вимагає значних інженерних навичок та умінь у областях: проектування та розробки ПЗ,

системному та мережевому програмуванні. Відповідно до поставлених вимог перед початком програмування системи необхідно описати наступне:

1. Формат роботи системи та її топологію.

2. Складові частини (класи, об’єкти, компоненти).

3. Взаємозв’язки між складовими частинами.

4. Процеси передачі даних у системі.

5. Механізми зберігання даних та читання конфігураційних файлів.

Додатково до вищезазначених системних вимог необхідно описати процеси та механізми реалізації функціональних вимог:

1. Реалізацію виявлення інтернет-загроз у веб-трафіку у режимі реального часу: базу даних сигнатур, алгоритм перевірки веб-трафіку, взаємопов’язані процеси, алгоритми дій у випадку виявлення інтернет-загроз.

2. Механізм блокування доступу користувачів до вказаних веб-ресурсів: процес створення конфігурації, сам механізм блокування, виведення повідомлення про блокування користувачу.

3. Процеси моніторингу мережевого трафіку: запис основних даних до БД.

На рис. 4.1 представлена загальна топологія та розташування вузлів у системі. Розроблювана система повинна знаходитись між хостами мережі для ефективного опрацювання наявності інтернет-загроз у веб-трафіку.

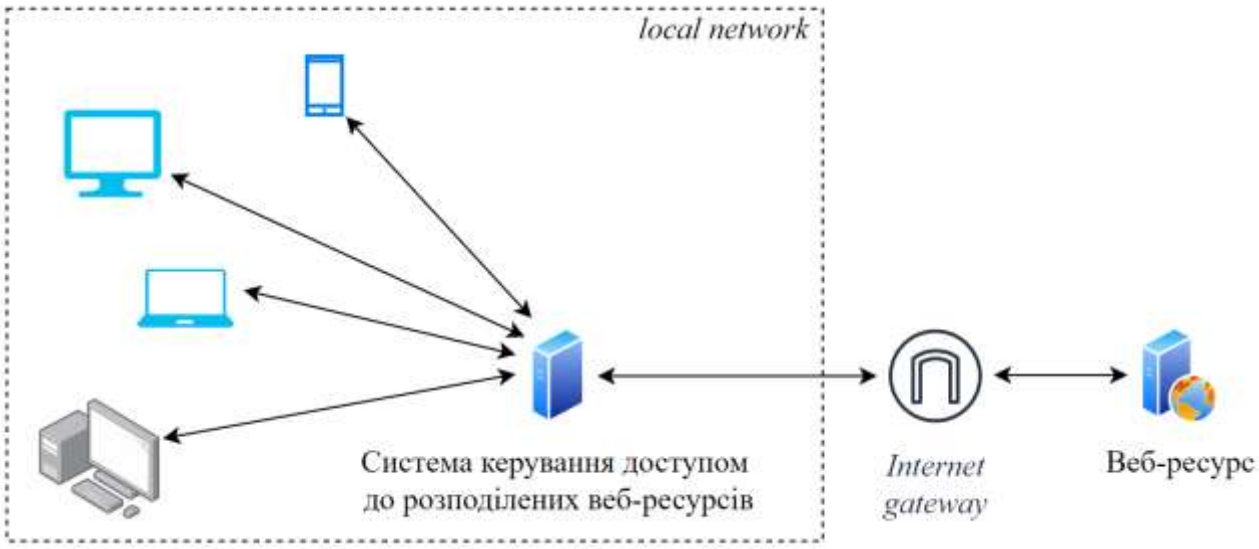

Рис.4.1. Топологія системи

На рис. 4.2 представлений компонентний склад системи. Кожен компонент відповідає за окремий функціонал, що логічно описує назва кожного компоненту. Ядро системи при цьому грає ключову роль.

Основні функції ядра системи:

1. Ініціалізація системи, завантаження конфігураційних файлів, створення та встановлення налаштувань компонентів.

2. Обробка та керування *TCP* з'єднаннями локальних клієнтів та зовнішніх веб-ресурсів.

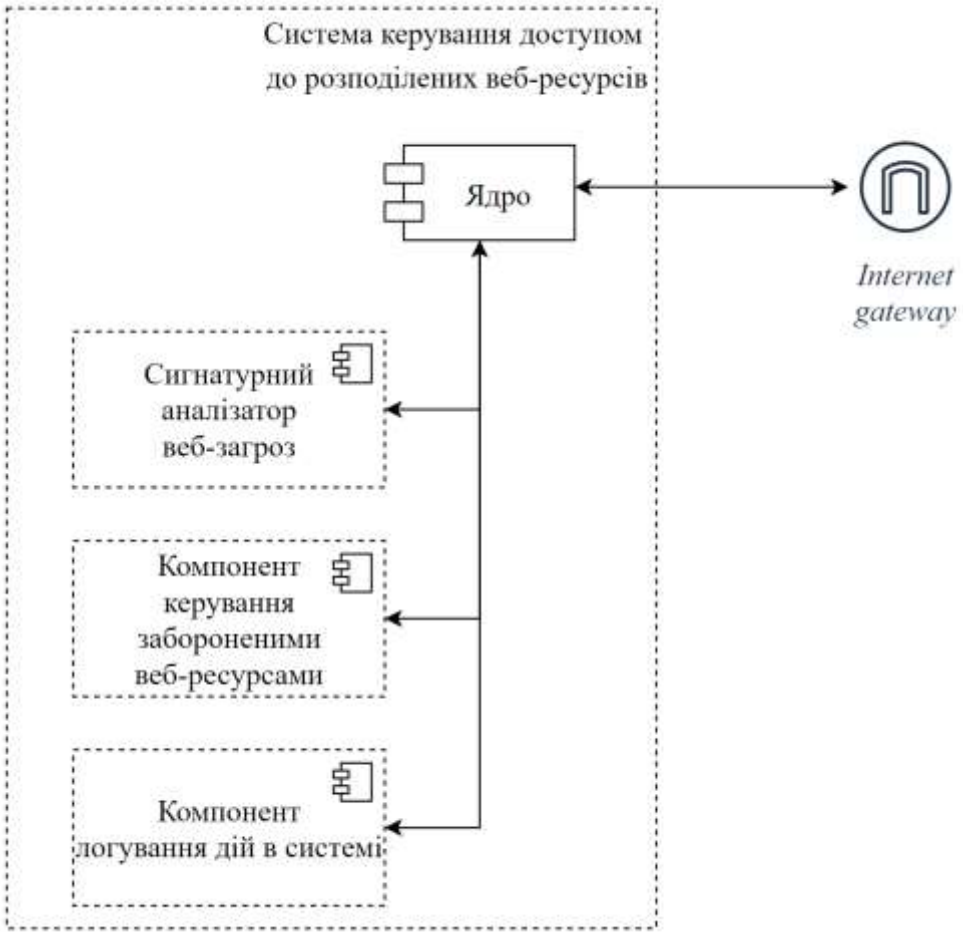

Рис.4.2. Компонентний склад системи керування доступом до розподілених веб-ресурсів

Розглянемо детальніше завдання ядра системи по обробці та керуванню *TCP*-з'єднаннями. Для створення з'єднання по протоколу *TCP/IP* використовуються наступні параметри:

1. *IP*-адреса та порт відправника.

2. *IP*-адреса та порт одержувача.

Відповідно для створення з'єднання з веб-сервером, який має необхідний веб-ресурс, клієнт повинен знати його *IP*-адресу та порт. Для знаходження цієї інформації клієнт звертається до зовнішнього *DNS*-серверу з запитом трансляції доменного імені у *IP*-адресу. При цьому використовувані порти у сучасному вебі – 80 для протоколу *HTTP* та 443 для протоколу *HTTPS*. Після процесу трансляції доменного імені та отримання усієї необхідної інформації клієнт встановлює *TCP* з'єднання з бажаним веб-сервером та формує повідомлення у форматі відповідно до протоколу *HTTP*. Клієнт записує сформоване повідомлення у канал з'єднання, а воно у свою чергу пересилається до веб-сервера низькорівневими мережевими механізмами, які не розглядаються у рамках дипломної роботи. Далі завданням веб-серверу є читання отриманого повідомлення з каналу з'єднання та його подальший аналіз. Після цього процесу веб-сервер "розуміє", що конкретно від нього вимагає клієнт, та відповідно задовольняє це прохання шляхом запису відповіді у канал з'єднання.

У рамках описаного процесу створювана система має виступати у ролі посередника між клієнтом та веб-сервером, а клієнт делегує описані вище обов'язки на систему. При цьому у системи з'являються повноваження впроваджувати додаткові функції у незалежний спосіб.

Тобто описаний процес містить етапи:

1. Встановлення *TCP* з'єднання.

2. Зчитування та запису інформації у *TCP* з'єднання.

3. Додатково створювана система повинна реалізовувати зчитування *HTTP* повідомлень від клієнтів з адресами одержувачів подальших повідомлень та встановлювати з'єднання з ними (працювати як *HTTP*-проксі).

Виконання вищезазначених етапів розподіляються між наступними класами (рис. 4.3):

1. *ProxyServerConnectionHandler*.

2. *ProxyServerConnectionReader*.

3. *ProxyServerConnectionWriter*.

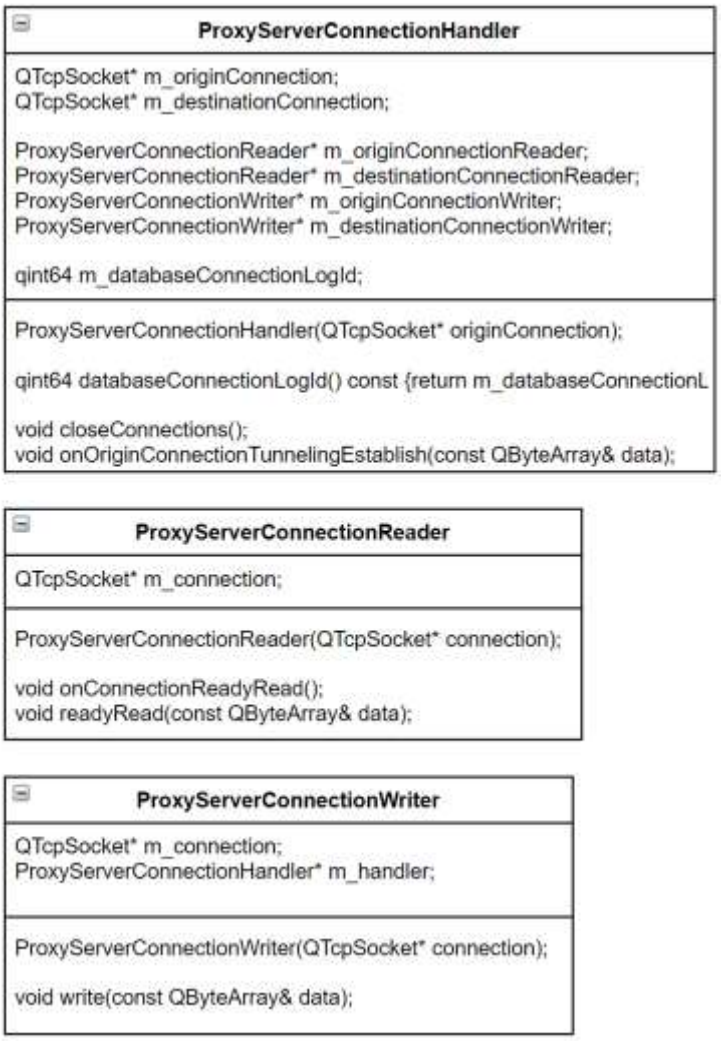

Рис.4.3. *UML*-діаграма класів відповідальних за обробку *TCP*-з’єднань

ProxyServerConnectionHandler володітиме двома об’єктами *TCP* з’єднань (з клієнтом та веб-сервером), по два об’єкта *ProxyServerConnectionReader* та *ProxyServerConnectionWriter* для відповідних зчитувань та записів для обох з’єднань, основну логіку для керування процесом з’єднань та допоміжний метод, що призначений для першочергової обробки запиту клієнта.

Розглянемо компонент сигнатурного аналізу загроз. Зазначені функції будуть покладені на класи (рис. 4.4):

1. *VirusSignatureProcessor*.

2. *VirusSignature*.

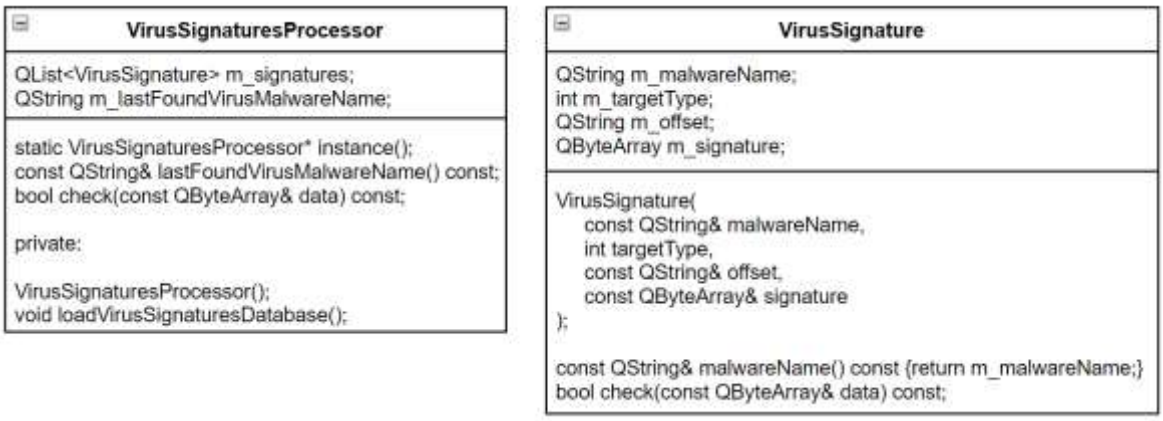

Рис.4.4. *UML*-діаграма класів сигнатурного аналізатору загроз

VirusSignatureProcessor використовуватиме патерн *Singleton* та завантажуватиме відомі сигнатури вірусів з заздалегідь підготовленої БД та створюватиме список об'єктів *VirusSignature*, які у свою чергу міститимуть операції по виявленню сигнатури вірусів у переданих даних. При цьому аналіз здійснюватиметься на кожному переданому пакеті *HTTP* даних.

Описання бази даних сигнатур вірусів. База даних сигнатур вірусів для виконання дипломної роботи була завантажена за посиланням [25]. База даних *Clamav* має формат *.cvd* та безкоштовно розповсюджується для усіх бажаючих. Формат файлу БД – архів *.tar.gz* за умови видалення перших 512 байтів, або ж його розпакування командою:

dd if=file.cvd bs=512 skip=1 | tar zxvf –

У архіві знаходяться різноманітні файли з розширеннями *.info, .ndb, .mdb, .hsb, .hdb*. Частина із них – це безпосередньо сигнатури вірусів закодовані у різних представленнях. Для виконання дипломної роботи була використана БД з розширенням *.ndb*. Формат даних, що знаходяться у цій БД – *Clamav Extended signature* (рис. 4.5):

MalwareName:TargetType:Offset:HexSignature

1. *MalwareName* – ім'я вірусу.
2. *TargetType* – тип файлів, що містять вірус.
3. *Offset* – типове місцеположення сигнатури вірусу у зараженому файлі.
4. *HexSignature* – безпосередня сигнатура вірусу у *HEX* форматі.

HexSignature під час ініціалізації системи перетворюватимемо у бінарний формат, оскільки уся інформація для аналізу буде у бінарному форматі.

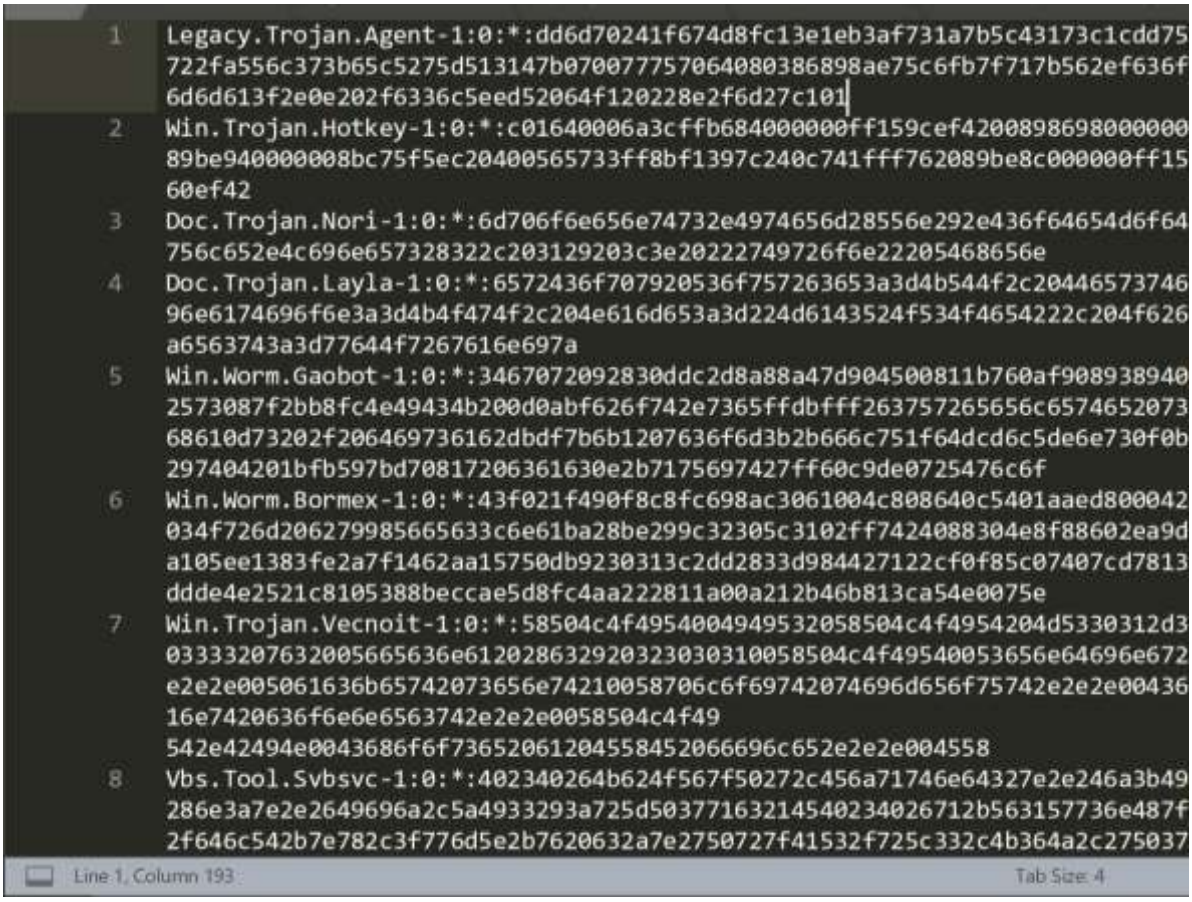

Рис.4.5. Формат *Clamav Extended signature*

Розглянемо компонент керування забороненими веб-ресурсами. До файлу налаштувань у форматі *.json* записуються доменні імена заборонених ресурсів (рис. 4.6), доступ до яких буде заборонено системою.

Рис.4.6. Конфігурація заборонених веб-ресурсів

При спробі доступу до зазначених у чорному списку ресурсів система одразу запише до *HTTP*-з'єднання з клієнтом відповідь з кодом 403 (*Forbidden*) та розірве з'єднання.

Основний клас, що відповідає за роботу цього механізму – *URLBlackListProcessor* (рис. 4.7). До його обов'язків входить зчитування файлу конфігурації при ініціалізації системи, зберігання *HTTP*-відповіді з кодом помилки та операція перевірки наданого *URL* на факт присутності у чорному списку.

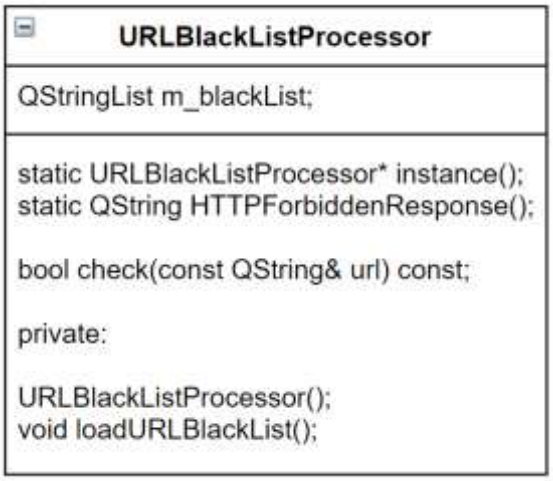

Рис.4.7. *UML*-діаграма класу *URLBlackListProcessor*

Розглянемо останній компонент системи – компонент логування дій. Логувати будемо наступне:

1. Створювані з'єднання у системі (*IP*-адреси та порти, час створення, час закриття з'єднання).

2. Кількість трафіку, що проходить через кожне з'єднання (пакети з'єднань та час їх створення).

3. Наявність вірусів у конкретному пакеті даних (ім'я вірусу).

Основний клас, що відповідатиме за збереження вищезазначених даних – *Database* (рис. 4.8).

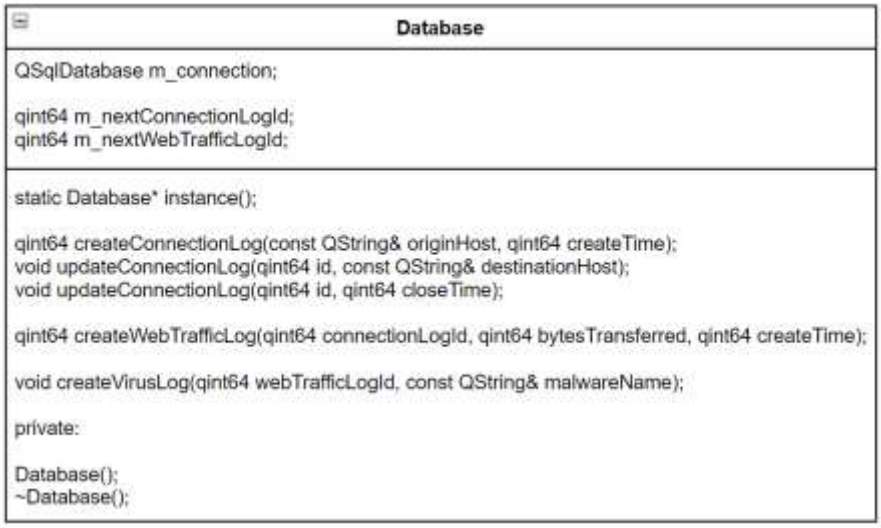

Рис.4.8. *UML*-діаграма класу *Database*

Зберігати дані будемо у БД *SQLite3*. Схема таблиць БД представлена на рис. 4.9.

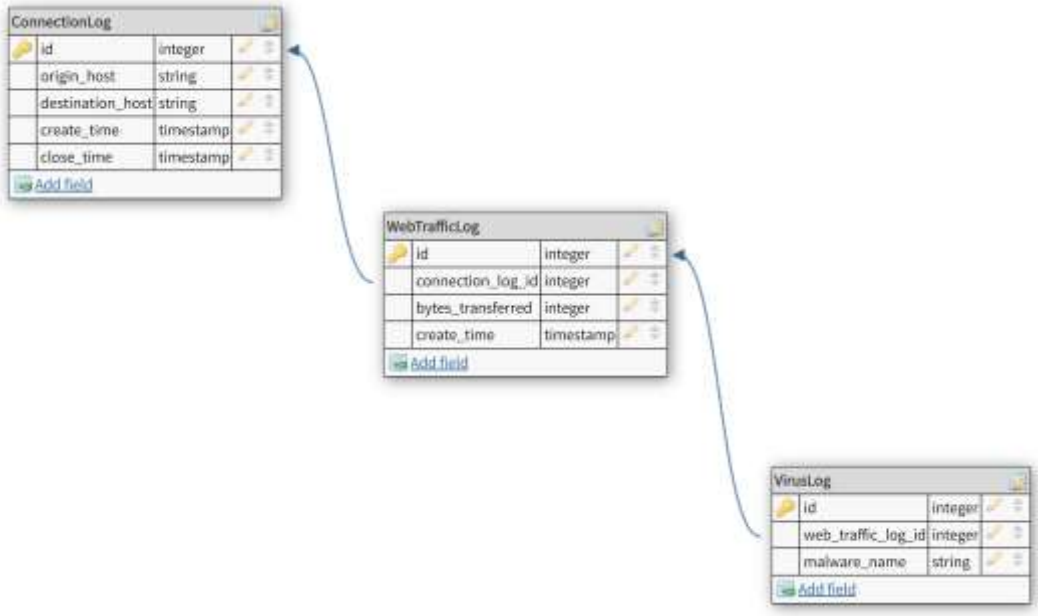

Рис.4.9. Схема таблиць БД для збереження логів системи

4.4. Результати розробки

Створена система керування доступом до розподілених веб-ресурсів призначена для роботи у локальній мережі з метою проміжного аналізу веб-трафіку та керування доступом до визначених веб-ресурсів.

Використані для розробки технології передбачають можливість застосування будь-якої ОС з підтримкою компілятора мови *C++*. Рекомендується використання ОС з сімейства *Linux*.

Фактично розроблена система працює як *HTTP*-проксі та використовує відповідний протокол для встановлення з'єднань з клієнтами. Щоб підключитися до системи необхідно знати *IP* та порт. Підключення може відбуватися як у межах окремих додатків (браузерів, програм тощо), що показано на рис. 4.10, так і у межах всієї ОС (рис. 4.11).

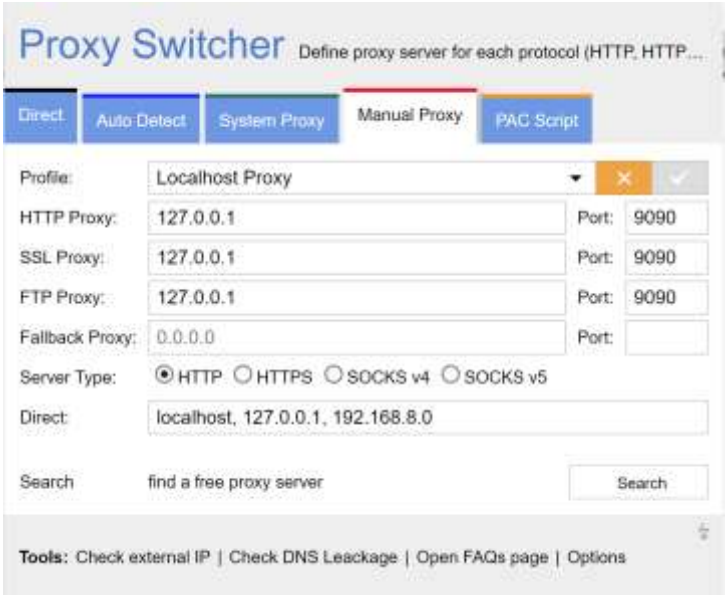

Рис.4.10. Налаштування параметрів підключення до системи з браузерного розширення

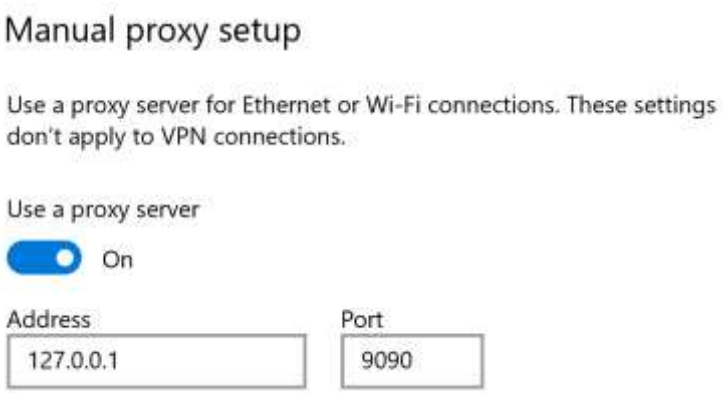

Рис.4.11. Налаштування параметрів підключення до системи з опцій *Windows* 10

Варто зазначити, що розгортання системи можливе також на локально та у глобальній мережі Інтернет, при цьому процес підключення до неї не змінюється.

До результатів роботи системи можна віднести ефективність у пошуку вірусних сигнатур. Вже у процесі тестування системи на звичайних сайтах були виявлені вірусні сигнатури (рис. 4.12).

Таблиця: VirusLog

	id	web_traffic_log_id	malware_name
	Фільтр	Фільтр	Фільтр
1	1	2538	Win.Spyware.Zbot-1290

Рис.4.12. Виявлена загроза у веб-трафіку

Результати блокування веб-ресурсу, наприклад *https://www.ukr.net*, представлені на рис. 4.13.

Рис.4.13. Результат блокування веб-ресурсу

Проведемо дослідження продуктивності системи – виміряємо швидкість обробки пакетів даних різного розміру при 100000 сигнатур вірусів. У результаті отримуємо, що одне апаратне ядро процесору *Intel Core I7* перевіряє найбільший пакет даних (65536 байтів) за ~500мс (рис. 4.14).

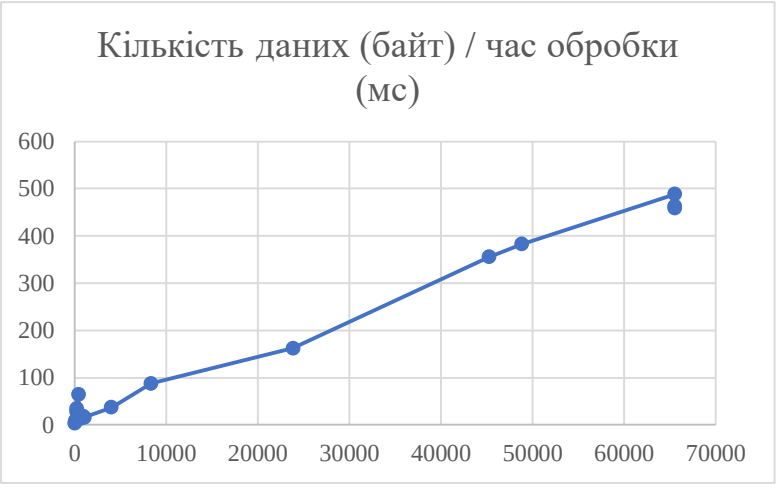

Рис.4.14. Відношення кількості даних до часу обробки на 100000 сигнатурах

Тобто затримка обміном мережевим трафіком через використання системи присутня, проте не є критичною. Для подальшого розвитку системи пропонується розпаралелення процесу сканування між декількома ядрами.

Висновки за розділом

У четвертому розділі розглянутий процес створення системи керування розподіленими веб-ресурсами: поставлені функціональні вимоги, обґрунтоване використання технологій для розробки, здійснено проектування системи, описаний процес розгортання та представлені результати роботи.

Перед створенням системи керування доступом до розподілених веб-ресурсів були з’ясовані функціональні вимоги: обробка з’єднань клієнтів, аналіз веб-трафіку на наявність сигнатур інтернет-загроз, конфігурація блокування веб-ресурсів, ведення журналу дій.

Поставлені вимоги визначили необхідні інженерні уміння для реалізації системи: мережеве та системне програмування, знання мережевих технологій та протоколів, навички роботи з БД. Відповідно до поставлених вимог було обґрунтовано використання інженерних інструментів: мова програмування, фреймворки, БД. Обраний інструментарій спростив процес створення системи та забезпечив кросплатформеність.

Було здійснено проектування системи: спроектовані класи за допомогою *UML*-діаграм, розроблена топологія системи, здійснений процес розбиття системи на окремі компоненти, представлений опис функціональних можливостей кожного компоненту системи, створена схема таблиць БД.

Після етапу проектування відбувся етап розробки. Усі поставлені функціональні вимоги реалізовані у повній мірі. Описаний процес розгортання системи, сфери її використання, представлені прикладні результати роботи кожного компоненту та зазначені їх основні параметри. Висловлені рекомендації щодо подальшого розвитку та удосконалення створеної системи.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено дослідження механізмів доступу до розподілених веб-ресурсів та розроблена система керування доступом до розподілених веб-ресурсів: проведений аналіз технологій та процесів розподілених веб-ресурсів, що включає в себе поняття “веб-ресурсу”, механізми обміну ними, додаткові механізми шифрування та стиснення даних при обміні веб-ресурсами; здійснено дослідження механізмів керування доступом до розподілених веб-ресурсів, що включає в себе мету керування ними, дослідження керівних впливів на процес доступу до веб-ресурсів, аналіз сучасних інтернет-загроз, проблеми та мету захисту комп’ютерних мереж, методи виявлення та інструментальні засоби для виявлення інтернет-загроз у комп’ютерних мережах; досліджені методи та засоби керування доступом до розподілених веб-ресурсів: описані технології проксі-серверів та веб-фільтрів, проведений порівняльний аналіз проксі-серверів, дослідженні протоколи проксі-серверів, розкритий вплив технологій шифрування та стиснення даних на процес аналізу веб-трафіку, розглянута статистична модель аналізу мережевого трафіку для виявлення інтернет-загроз методом сигнатурного аналізу у режимі реального часу, а на її основі створений алгоритм аналізу наявності інтернет-загроз у веб-трафіку; проведена розробка системи керування доступом до розподілених веб-ресурсів: сформовані функціональні вимоги, обґрунтоване використання технологій, здійснене проектування та програмування системи, описаний процес розгортання системи та представлені результати розробки.

До початку процесу створення системи були сформовані функціональні вимоги. У результаті роботи реалізоване наступне:

1. Система здійснює аналіз вхідного мережевого трафіку на наявність інтернет-загроз (вірусів, хробаків, троянів тощо) методом сигнатурного аналізу.

2. Система може блокувати доступ до веб-ресурсів з можливістю ручного налаштування списку таких веб-ресурсів.

3. Система постійно здійснює моніторинг мережевого трафіку та веде записи по основним мережевим характеристикам кожного вузла у окрему БД.

Усі поставлені функціональні мірі реалізовані та працюють у повній мірі.

Додатково до вищеописаних функцій система виконує:

1. Встановлення зв’язку з клієнтами за допомогою механізмів *HTTP*-проксі.

2. Здійснення *HTTP*-запитів на користь клієнтів для отримання веб-ресурсів та пересиланню їх клієнтам.

3. Формує взаємозв’язки між усіма компонентами системи та виконує їх ініціалізацію.

Впровадження створеної системи керування доступом до розподілених веб-ресурсів можливе у локальній мережі та глобальній мережі Інтернет. Основними вимогами до розгортання системи та впровадження її у роботу є наявність достатньої кількості обчислювальних ресурсів та широкого інтернет-каналу.

У результаті дипломної роботи виконані наступні задачі:

1. Досліджені можливі керівні впливи на процес керування доступом до розподілених веб-ресурсів.

2. Проведено дослідження виявлення інтернет-загроз при доступі до веб-ресурсів.

3. Досліджені методи та інструментальні засоби для виявлення інтернет-загроз при доступі до розподілених веб-ресурсів.

4. Розкрито поняття веб-ресурсу: суть поняття, описані основні механізми обміну веб-ресурсами, розкриті процеси додаткової обробки веб-ресурсів: шифрування та стискання даних.

5. Проведено дослідження проксі-серверів та використані технології HTTP-проксі у створеній системі керування доступом до розподілених веб-ресурсів.

6. Обґрунтовано вибір технологій для розробки системи керування доступом до розподілених веб-ресурсів.

7. Спроектовано систему керування доступом до розподілених веб-ресурсів: встановлені функціональні вимоги, проведено розбиття системи на компоненти, спроектовані класи та схеми БД.

8. Розроблено систему керування доступом до розподілених веб-ресурсів, надані рекомендації щодо її розгортання та використання, представлені результати тестування системи.

Перший розділ розкрив питання аналізу технологій та процесів розподілених веб-ресурсів. З'ясовано, що таке веб-ресурс, які механізми та протоколи використовуються для їх отримання та створення доступу до них. Детально описані деякі частини протоколу *HTTP*, що використовувалися у подальшій роботі. Додатково проаналізована тема шифрування та стискання даних веб-ресурсів перед їх передачею по мережі.

Другий розділ розкрив тему механізмів керування доступом до розподілених веб-ресурсів. З'ясована мета керування розподіленими веб-ресурсами, що полягає у здійснені керуючих впливів на процес обміну веб-ресурсами для блокування визначених веб-ресурсів, виявленні інтернет-загроз у веб-трафіку та постійного моніторингу *HTTP*-з'єднань. Досліджені можливі керівні впливи на процес керування доступом до розподілених веб-ресурсів: створення та розривання *HTTP*-сесій, аналіз та/або редагування вмісту тіла *HTTP*-повідомлень, редагування формату даних *HTTP*-повідомлень, блокування та/або перенаправлення *HTTP*-трафіку. Обрані керівні впливи для подальшого дослідження: аналіз вмісту тіла *HTTP*-повідомлень на наявність певних інтернет-загроз, блокування *HTTP*-трафіку для визначених веб-ресурсів, редагування формату даних *HTTP*-повідомлень для їх підготовки до подальшого аналізу. Проаналізовані сучасні інтернет-загрози та зроблений висновок про зв'язок теми аналізу та виявлення інтернет-загроз у веб-трафіку. У подальшому було з'ясовано, що постійне зростання користувачів Інтернетом спонукає створення шкідливого ПЗ для крадіжок та знищення даних користувачів. Сформований висновок, що будь-яка система захисту користувачів повинна бути максимально "прозора" та не вносити додаткового навантаження на ресурси системи цільового користувача. Перелічені основні методи та інструментальні засоби для виявлення інтернет-загроз у комп'ютерних мережах: сигнатурний аналіз, евристичний аналіз, продукційні системи, статистичний метод, штучні нейронні мережі. Описані типи та алгоритми роботи вірусів.

Третій розділ розкрив питання методів та засобів керування доступом до розподілених веб-ресурсів. Окрема увага була приділена проксі-серверам: сферам їх використання, призначенню, технологіям та протоколам. Проведений аналіз сучасних веб- та проксі-серверів: *Apache* та *NGINX*. Здійснене дослідження протоколів проксі: *Socks* 4, *Socks* 5, *HTTP*-proxy та зроблений висновок щодо використання механізмів *HTTP*-проксі у розроблюваній системі керування доступом до розподілених веб-ресурсів. Проведене дослідження впливу технологій шифрування та стиснення даних на процес аналізу веб-трафіку. З'ясовано, що вищезазначені технології покращують досвід користувачів, проте значно ускладнюють процеси аналізу веб-трафіку. Описана статистична модель виявлення мережевих вторгнень, що базується на сигнатурному аналізі. На базі цієї моделі запропонований алгоритм дій для виявлення інтернет-загроз у веб-трафіку у режимі реального часу.

У четвертому розділі описані процеси створення системи керування доступом до розподілених веб-ресурсами. Сформовані основні функціональні вимоги системи. Здійснене проектування ПЗ, що включило в себе етапи опису класів системи у вигляді *UML*-діаграм, опису механізмів роботи та призначення компонентів, створення схем баз даних для зберігання даних тощо. Перед етапом проектування обґрунтовані використовувані технології для програмування та реалізацій функцій системи. На базі обраного інструментарію було здійснене програмування системи керування доступом до розподілених веб-ресурсів. Заздалегідь визначена топологія системи окреслює можливий варіант розгортання та впровадження системи, проте не обмежує інші: на локальному комп'ютері, у глобальній мережі тощо.

Остаточним результатом роботи стало проведене дослідження у сфері керування доступом до розподілених веб-ресурсів та у сфері аналізу веб-трафіку на наявність інтернет-загроз, а також створення системи керування доступом до розподілених веб-ресурсів.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *RFC 1945. Hypertext Transfer Protocol – HTTP/1.0* [Інтернет-ресурс]. – Режим доступу: *https://tools.ietf.org/html/rfc1945* вільний.

2. *RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [Інтернет-ресурс]. – Режим доступу: *https://tools.ietf.org/html/rfc7231* вільний.

3. *RFC 7540 - Hypertext Transfer Protocol Version 2 (HTTP/2)* [Інтернет-ресурс]. – Режим доступу: *https://tools.ietf.org/html/rfc7540* вільний.

4. Шелухин О.И. Обнаружение вторжений в компьютерные сети (сетевые аномалии) / О.И. Шелухин, Д.Ж. Сакалема, А.С. Филинова. – М.: Горячая линия – Телеком, 2013. – 221 с.

5. *Denning D. An intrusion-detection model. In Proc. // IEEE Symposium on Security and Privacy. - 1987. - Vol. 13, No 2. – P. 222-232.*

6. *Sheyner O. Scenario Graphs and Attack Graphs. / PhD thesis, SCS, Pittsburgh: Carnegie Mellon University. – 2004. – P. 141.*

7. *Edward G. Intrusion Detection. 1st ed., Intrusion.Net Books / New Jersey: Sparta. – 1999. – P. 218.*

8. *Eckmann S.T., Vigna G., Kemmerer R. A. STATL: An Attack Language for State-based Intrusion Detection // Dept. of Computer Science, University of California, Santa Barbara. – 2000. – Vol. 12, No 2. – P. 71-103.*

9. *Vigna G., Kemmerer R. A. NetSTAT: A Network-based Intrusion Detection Approach // Proceedings of the 14th Annual Computer Security Application Conference. – 2000. – P. 73-81.*

10. Гамаюнов Д.Ю., Смелянский Р.Л. Модель поведения сетевых объектов в распределенных вычислительных системах / Д.Ю. Гамаюнов, Р.Л. Смелянский // Программирование. – 2007. – № 4. – С. 20–31.

11. *Lee W., Stolfo S. Data mining approaches for intrusion detection. // In Proc. of the 7th USENIX Security Symposium. – 1998. – No 7. – P. 79-94.*

12. Гаврилов А.В. Применение постоянно модифицирующихся нейронных сетей для защиты программного обеспечения / А.В. Гаврилов // Нейрокомпьютеры: разработка, применение. – 2008. – № 1-2. – С. 90-101.

13. Оладько В.С. Причины и источники сетевых аномалий / В.С. Оладько, С.Ю. Микова, М.А. Нестеренко, Е.А. Садовник // Молодой ученый. – 2015. – №22. – С. 158-161.

14. Басараб М.А., Строганов И.С. Обнаружение аномалий в информационных процессах на основе мультифрактального анализа / М.А. Басараб, И.С. Строганов // Вопросы кибербезопасности. – 2014. – №4(7). – С. 30-40.

15. *Debar H. Towards a taxonomy of intrusion detection systems / H. Debar, H. Dacier, A. Wespi // Computer Networks. – 1999. – № 31. – P. 805-822.*

16. Гатаулин С.И. Распознавание вирусов с частичной полиморфностью / С.И. Гатаулин. – Москва, 2012. – 51 с.

17. Абалмазов Э.И. Методы и инженерно-технические средства противодействия информационным угрозам / Э.И. Абалмазов. – М.: Гротек, 1997. – 248 с.

18. Бил Дж. Обнаружение вторжений. / Дж. Бил. – Москва, 2006. – 656 с.

19. Кораблев Н.М. Нейросетевой эвристический анализатор вредоносных программ с иммунным обучением / Н.М. Кораблев, М.В. Кушнарев, Д.П. Ужвий // Радиоэлектроника и информатика: научно-техн.журнал. – 2014. – № 2 (65). – С. 19-25.

20. Большев А.К., Лисс А.Р. Прототип эвристической системы обнаружения вторжений в компьютерные сети на основе метода главных компонент / А.К. Большев, А.Р. Лисс // Научно-Технические Ведомости СПбГПУ, Серия «Информатика, Телекоммуникации, Управление». – 2010. – Т. 4(103). – С. 200-205.

21. Балакін С.В. Методи та засоби підвищення достовірності ідентифікації несанкціонованих дій та атак в комп’ютерній мережі : дис. … канд. техн. наук / Балакін С.В. ; Нац. авіа. ун-т. – 2018. – 151 с.

22. *Vere S.A. Relational production systems // Artificial Intelligence. – 1977. – №8. – P. 47-68.*

23. Медведев Н.В. Применение метода статического сигнатурного анализа для выявления дефектов безопасности веб-приложений / Н.В. Медведев, А.С. Марков, А.А. Федин // Наука и образование: электронное научно-техническое издание. – 2012. – № 9. – С. 21-31.

24. *Golovko V. Neural Networks approaches for Intrusion Detection and Recognition // Computing. – 2006. – Vol. 5. № 3. – P. 118-125.*

25. *ClamavNet. Source Code* [Інтернет-ресурс] / Web-сайт: *Clamav*. – Режим доступу: *https://www.clamav.net/downloads* вільний.

26. *Shivani S. Virus Detection using Artificial 137 Neural Networks / S. Shivani, J. Himali, S. Sathvik, B. Kiran // International Journal of Computer Applications. – 2013. – Vol. 84. № 5. – P. 17-33.*

27. Милославская Н.Г. Интрасети: обнаружение вторжений / Н.Г. Милославская, А.И. Толстой – М.: ЮНИТИДАНА, 2001.

28. Пелешенко В.С. Обзор методик обнаружения сетевых атак // Материалы II Международной научно-технической конференции «Инфокоммуникационные технологии в науке и технике», ч. II. – 2006.

29. Чипига А.Ф., Пелешенко В.С. Обзор моделей систем обнаружения атак в ЛВС и выявление их недостатков / А.Ф. Чипига, В.С. Пелешенко // Материалы II Международной научно-технической конференции «Инфокоммуникационные технологии в науке и технике», ч. II. – 2006.

30. Кудряшов И.С. Регистрация событий в системах обнаружения компьютерных атак // Материалы VII Международной научно-практической конференции «Информационная безопасность». - Таганрог: Изд-во ТРТУ, 2005.

31. Пелешенко В.С. Математическая модель процессов связи узлов в сети при обнаружении и предотвращении несанкционированного доступа к информации // Материалы 9-й региональной научно-технической конференции «Вузовская наука – Северо-Кавказскому региону». – 2005.

32. *Hrushak S., Pavlenko C. Advantages of DNS-over-HTTPS over DNS / S. Hrushak, C. Pavlenko // Computer and information systems and technologies (Kharkiv, Ukraine, 2020). – 2020.*

33. *Hrushak S., Pavlenko C. Virtual Private Network and its use in secured corporative networks / S. Hrushak, C. Pavlenko // Computer and information systems and technologies (Kharkiv, Ukraine, 2019). – 2019.*