МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.53

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Методи виявлення та протидії DDoS-атакам на інформаційні системи

Виконавець: Дмитро БЕЗГУБЕНКО
Керівник: д.т.н., професор Сергій ТОЛЮПА
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: д.т.н., професор Сергій ТОЛЮПА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«___» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Безгубенка Дмитра Вікторовича

1. Тема: *Методи виявлення та протидії DDoS-атакам на інформаційні системи* затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: проаналізувати нормативно-правову базу щодо забезпечення національної безпеки України в інформаційній сфері від DDoS-атак, розкрити сутність поняття DDoS-атак та їх видів, визначити механізми реалізації DDoS-атак та з'ясувати методи захисту від них, висвітлити вплив DDoS-атак на веб-ресурси; здійснити аналіз мови програмування та середовища розробки, що необхідні для ефективної розробки програмного продукту для протидії DDoS-атак; на основі проведених досліджень здійснити проектування та розробку програмного продукту для ефективного захисту від DDoS-атак.
4. Зміст пояснювальної записки: теоретичні аспекти DDoS-атак; огляд інструментальних засобів розробки; проектування та розробка програми захисту від DDoS -атак.

5. КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	20.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	25.10.2023	*Виконано*
4.	Збір інформації	30.10.2023	*Виконано*
5.	Дослідження теоретичних аспектів DDoS-атак	06.11.2023	*Виконано*
6.	Огляд інструментів і технологій розробки системи для протидії DDoS-атак	13.11.2023	*Виконано*
7.	Проектування та розробка програми захисту від DDoS -атак	20.11.2023	*Виконано*
8.	Апробація роботи на міжнародній науково-практичній конференції «ЖИВУЧІСТЬ ТА РЕЗИЛЬЄНТНІСТЬ – 2023»	19.10.2023	*Виконано*
9.	Перевірка на антиплагіат	11.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	14.12.2023	*Виконано*
11.	Оформлення презентації	18.12.2023	*Виконано*
12.	Отримання рецензій	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Дмитро БЕЗГУБЕНКО

(підпис, дата)

Керівник кваліфікаційної роботи Сергій ТОЛЮПА

(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел, одного додатку і має 78 сторінок основного тексту, 15 рисунків, 7 сторінок додатків. Список використаних джерел містить 58 найменувань і займає 6 сторінок. Загальний обсяг роботи 85 сторінок.

Метою роботи є розробка власної реалізації програмного продукту, спрямованого на захист від атак типу DDoS.

В роботі вирішено задачу побудови власної реалізації програмного продукту, спрямованого на захист від атак типу DDoS..

В роботі розроблено алгоритм та програмне забезпечення для ефективного захисту від DDoS-атак. Основна мета розробленої програми полягає у реалізації системи захисту від DDoS-атак та одночасно наданні можливості редагування та збереження списку IP-адрес.

Розроблений програмний продукт щодо протидії DDoS-атакам на інформаційні системи є важливим в сучасному інформаційному середовищі. Він може забезпечити надійність, безпеку та стійкість різноманітних інформаційних систем, зокрема веб-ресурсів, від атак типу DDoS.

Ключові слова: DDOS-АТАКА, ВЕБ-РЕСУРСИ, IP-АДРЕСА, ДАНІ, ІНФОРМАЦІЯ, ЗАХИСТ, МЕТОД, КОРИСТУВАЧ, БЕЗПЕКА.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому світі інформаційні системи стали ключовою складовою успішного функціонування бізнесу, державних установ, та суспільних структур загалом. Із зростанням ролі технологій у нашому житті, з'явилася також загроза їхньому функціонуванню - кібератаки. Однією з найпоширеніших та найагресивніших форм кібератак є розподілені атаки з відмовою обслуговування, або DDoS-атаки. Ці атаки спрямовані на недоступність веб-ресурсів та інформаційних систем, завдаючи серйозних фінансових та репутаційних збитків.

DDoS-атаки є складними та небезпечними, оскільки вони використовують розподілені мережі зомбі-комп'ютерів для масштабного завантаження веб-серверів чи мережевої інфраструктури. Внаслідок цього, легітимні користувачі не можуть отримати доступ до важливих веб-ресурсів, що може призвести до фінансових втрат, втрати клієнтів та навіть порушенням безпеки.

Забезпечення захисту веб-ресурсів від DDoS-атак є критично важливим завданням для підтримання надійності та доступності інформаційних систем. Порушення роботи цих систем може мати серйозні наслідки, особливо в сферах, де доступ до даних і сервісів є життєво важливими, таких як фінансовий сектор, охорона здоров'я, освіта та громадська безпека.

Метою кваліфікаційної роботи є розробка власної реалізації програмного продукту, спрямованого на захист від атак типу DDoS.

Для досягнення поставленої мети передбачається виконання наступних **задач**:

– проаналізувати нормативно-правову базу щодо забезпечення національної безпеки України в інформаційній сфері від DDoS-атак, розкрити сутність поняття DDoS-атак та їх видів, визначити механізми реалізації DDoS-атак та з’ясувати методи захисту від них, висвітлити вплив DDoS-атак на веб-ресурси;

– здійснити аналіз мови програмування та середовища розробки, що необхідні для ефективної розробки програмного продукту для протидії DDoS-атак;

– на основі проведених досліджень здійснити проектування та розробку програмного продукту для ефективного захисту від DDoS-атак.

Галузь застосування. Розроблений програмний продукт щодо протидії DDoS-атакам на інформаційні системи є важливим в сучасному інформаційному середовищі. Він може забезпечити надійність, безпеку та стійкість різноманітних інформаційних систем, зокрема веб-ресурсів, від атак типу DDoS.

Об'єктом дослідження є процес виявлення та протидії DDoS-атакам на інформаційні системи, зокрема веб-ресурси.

Предметом дослідження є програмний продукт протидії DDoS-атакам.

Методи дослідження. В процесі дослідження використовувалися основні методи такі, як аналіз (для вивчення сучасних методів виявлення та протидії DDoS-атакам, а також огляд існуючих рішень у цій галузі), моделювання, експеримент та спостереження (для розробки програмного продукту спрямованого на ефективний захист від DDoS-атак).

Новизна одержаних результатів полягає в наступному:

– розроблена програма захисту поєднує в собі веб-сервер Flask для відображення інформації на веб-інтерфейсі та Tkinter для реалізації текстового редактора. Додатково, вона використовує бібліотеки для обмеження частоти запитів, що робить її відмінною уніфікованою системою для виявлення та захисту від DDoS-атак.

– розроблена програма автоматично веде список IP-адрес, які спробували викликати DDoS-атаку. Це дозволяє вести детальний журнал подій та аналізувати характеристики атаки, що спрощує подальше реагування на можливі загрози

– вперше реалізована система тимчасового обмеження IP-адрес, яка дозволяє автоматично визначати та обмежувати кількість запитів від конкретного IP за певний період. Це допомагає зменшити вплив DDoS-атак та запобігти надмірному завантаженню системи.

Практичне значення отриманих результатів полягає у наступному:

– розроблена програма має практичне застосування для виявлення та протидії DDoS-атакам на інформаційні системи. Захисна система автоматично реагує на підозрілу активність, забезпечуючи стабільну роботу інформаційних ресурсів.

– можливість динамічного ведення списку IP-адрес та системи тимчасового обмеження надає засоби для докладного аналізу інцидентів. Це важливо для вдосконалення стратегій виявлення та захисту в майбутньому.

– інтеграція Flask та Tkinter робить програму зручною у використанні та дозволяє адміністраторам ефективно взаємодіяти з системою, що спрощує процес виявлення та відповіді на DDoS-атаки.

Апробація. Основні положення роботи доповідалися та обговорювалися на Міжнародній науково-практичній конференції «ЖИВУЧІСТЬ ТА РЕЗИЛЬЄНТНІСТЬ – 2023», м. Київ, 19 жовтня 2023 р., ІПМЕ ім. Г.Є. Пухова НАН України.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ АСПЕКТИ DDOS-АТАК

1.1 Аналіз нормативно-правової бази кібербезпеки

У світлі швидкого розвитку комп'ютерних технологій, боротьба з DDoS-атаками стає надзвичайно важливою. Особливо актуальною ця проблема стає, коли ми говоримо про захист інформаційно-телекомунікаційних систем, що використовуються органами державної влади, правоохоронними органами, військовими формуваннями та об'єктами критичної інфраструктури. Інформаційні ресурси цих систем призначені для задоволення життєво важливих потреб громадян, особи, суспільства та держави в цілому.

Нормативно-правова база щодо забезпечення національної безпеки України в інформаційній сфері від DDoS-атак має певну ієрархію нормативних актів.

На вищому рівні ієрархії знаходяться норми Конституції України [1], що закріплює концептуальні положення національної безпеки України в усіх сферах її існування, а також Указ Президента України №392/2020 «Про Стратегію національної безпеки України» [2]. Ці документи враховують основні положення міжнародних договорів і угод, які були ратифіковані Україною і мають стосунок до її національної безпеки.

Стратегія національної безпеки України [2] (далі – Стратегія) зосереджує увагу на двох аспектах: кібербезпеці та протидії пропаганді країни-агресора. В рамках Стратегії особлива увага приділяється деструктивним інформаційним кампаніям росії, які викликають протиріччя у суспільстві та спрямовані на розпалювання ворожнечі, сприяння конфліктам та роз'єднанню суспільства. Автори Стратегії вважають, що важливими діями для подолання цієї загрози є формування єдиної інформаційної політики держави та посилення стратегічних комунікацій.

У секторі безпеки і оборони держава має плани завершити створення національної системи кібербезпеки і розвивати сучасні здібності для суб'єктів, які забезпечують кібербезпеку та кібероборону. Одною з ключових цілей даного сектору є забезпечення супротивника неприйнятними втратами у кіберпросторі.

Зокрема, Стратегія [2] визначає також невійськові загрози, які виникають з боку росії, та пропонує наступні заходи для їх подолання:

– боротьба з розвідувально-підривною діяльністю, спеціальними інформаційними операціями і кібератаками, зокрема з DDoS-атаками, а також з підривною пропагандою з боку росії;

– запобігання, виявлення та ліквідація проявів сепаратизму, тероризму, екстремізму, ліквідація незаконних збройних угруповань, політичного насильства та інших порушень конституційного порядку;

– здійснення збору повної та чнадійної інформації з попереджувальною метою щодо ситуації в Україні та у світі, протидія зовнішнім загрозам національній безпеці України та підтримання національних інтересів України.

На другому рівні даної ієрархії знаходяться закони конститутивного напрямку, які визначають важливі положення та встановлюють вимоги для забезпечення національної безпеки в інформаційній сфері. Серед них можна виділити Положення № 1426 «Про організаційно-технічну модель кіберзахисту», що було затверджено постановою КМУ 29.12.2021 р. [3], а також «Загальні вимоги до кіберзахисту об’єктів критичної інфраструктури» №518, що були затверджені постановою КМУ 19.06.2019 р. [4].

Організаційно-технічна модель кіберзахисту [3] - це комплекс заходів, ресурсів і інструментів, спрямованих на ефективне реагування на кібератаки, зокрема на DDoS-атаки, та кіберінциденти, а також на впровадження протидійних заходів з метою зменшення вразливості комунікаційних систем.

Загальні вимоги [4] визначають обов'язкові умови, які стосуються організаційно-методологічних, технічних та технологічних аспектів кіберзахисту об'єктів критичної інфраструктури. Ці умови повинні бути виконані підприємствами, установами та організаціями, що підпадають під визначення

об'єктів критичної інфраструктури згідно з чинним законодавством. Впровадження заходів з кіберзахисту дозволить цим підприємствам, установам та організаціям забезпечити захист від кібератак, зокрема віл DDoS-атак, запобігти порушенню конфіденційності, цілісності та доступності їхніх інформаційних ресурсів і забезпечити безперебійну роботу об'єктів критичної інфраструктури.

На третьому рівні ієрархії розташовані закони України інституційного характеру, в яких закріплені основні принципи діяльності державних органів у процесі забезпечення національної безпеки в інформаційній та інших сферах життєдіяльності громадянина, суспільства та держави. До цього рівня належать, наприклад, Закон України № 2163-VIII «Про основні засади забезпечення кібербезпеки України» від 05.10.2017 р. [5] та інші подібні нормативні акти.

Закон України «Про основні засади забезпечення кібербезпеки України» [5] встановлює юридичні та організаційні принципи для захисту життєво важливих інтересів людини і громадянина, суспільства та держави, а також національних інтересів України в кіберпросторі. Він визначає основні цілі, напрями та принципи державної політики у галузі кібербезпеки, регулює повноваження державних органів, підприємств, установ, організацій, громадян та осіб у цьому контексті, і встановлює основи координації їхньої діяльності для забезпечення кібербезпеки.

Отже, аналіз чинного законодавства дозволяє нам зробити висновок про наявність достатньої нормативно-правової бази щодо перспектив захисту національної безпеки України, зокрема в інформаційній сфері, від DDoS-атак.

1.2 Поняття DDoS-атак та їх види

DDoS-атаки (розподілені атаки на відмову в обслуговуванні) [6] – це форма кібератак, яка полягає в спробі перевантажити або знеструмити цільовий сервер або мережу, надсилаючи велику кількість запитів або трафіку.

Ці атаки можуть бути здійснені різними способами, тому існують різні види DDoS-атак. Ось деякі з них:

1. Атака на основі великого обсягу трафіку (Volumetric Attack).
2. Атака на ресурси (Resource Exhaustion Attack).
3. Атака на застосунок (Application Layer Attack).
4. Атака на пропускну здатність (Bandwidth Attack).
5. Атака на протоколи (Protocol Attack).
6. Змішані атаки (Hybrid Attacks).

Атака на основі великого обсягу трафіку [7] – це один із видів розподілених атак на відмову в обслуговуванні, яка спрямована на перевантаження пропускної здатності цільового сервера або мережі шляхом надсилання надмірної кількості трафіку. Цей вид атаки стає все більш поширеним і загрожує доступності веб-сайтів та інших мережевих ресурсів. До атак на основі великого обсягу трафіку включають:

– ICMP флуд – за даного виду атак зловмисники надсилають велику кількість ICMP (Internet Control Message Protocol) запитів, таких як пінги, до цільового сервера або мережі, що призводить до перевантаження і недоступності;

– UDP флуд – за даного виду атак зловмисники надсилають велику кількість запитів через UDP (User Datagram Protocol) до цільового сервера або мережі, але оскільки UDP не вимагає встановлення з'єднання, це може спричинити перевантаження ресурсів і недоступність;

– TCP SYN/ACK флуд – за даного виду атак зловмисники надсилають велику кількість запитів на встановлення з'єднання (SYN) або підтвердження завершення з'єднання (ACK) до сервера, в результаті це може переповнити черги обробки сервера та зробити його недоступним для легітимних користувачів;

– HTTP флуд – за даного виду атак зловмисники надсилають велику кількість HTTP запитів до веб-сервера, що в результаті може призвести до перевантаження сервера та відмови в обслуговуванні для легітимних користувачів;

– DNS ампліфікація – за даного виду атак зловмисники використовують велику кількість запитів до DNS-серверів, використовуючи DNS-сервери як посередників для збільшення обсягу атаки, в результаті дана атака може бути особливо ефективною через великий обсяг DNS-відповідей, які генеруються відповідь на невелику кількість запитів;

– NTP ампліфікація – за даного виду атак зловмисники використовують Network Time Protocol (NTP) для збільшення обсягу атаки, надсилаючи масиви запитів до NTP-серверів;

– SSDP ампліфікація – за даного виду атак зловмисники використовують Simple Service Discovery Protocol (SSDP) для збільшення обсягу атаки, викликаючи велику кількість відповідей від SSDP-серверів на невелику кількість запитів.

Атака на ресурси [8] – це форма розподіленої атаки на відмову в обслуговуванні, яка спрямована на вичерпання обчислювальних, мережевих або інших ресурсів цільового сервера або мережі. Ця атака не вимагає великого обсягу трафіку, але може спричинити серйозну відмову в обслуговуванні, оскільки вона намагається вичерпати ресурси, які сервер або мережа можуть виділити для легітимних користувачів. До атак на ресурси включають:

– CPU Exhaustion Attack – у цьому виді атаки зловмисники намагаються виконувати велику кількість обчислень або запускати процеси, які споживають багато процесорного часу, що в результаті може призвести до тимчасового перевантаження CPU і сповільнення роботи системи;

– Memory Exhaustion Attack – у цьому виді атаки зловмисники намагаються заповнити всю доступну оперативну пам'ять системи або інших ресурсів пам'яті, таких як диск, що в результатіможе призвести до витоку пам'яті або відмови в обслуговуванні;

– Disk Space Exhaustion Attack – у цьому виді атаки зловмисники намагаються заповнити весь доступний дисковий простір системи, що може призвести до відмови у зберіганні даних або витоку інформації;

– Thread/Connection Exhaustion Attack – у цьому виді атаки зловмисники можуть створювати велику кількість одночасних з'єднань або потоків, доступних для системи, щоб вичерпати цей ресурс і унеможливити нові підключення.

Атака на застосунок [9] – це форма розподіленої атаки на відмову в обслуговуванні, за якої зловмисники намагаються знеструмити конкретний веб-застосунок або службу, надсиланням великої кількості запитів, які вимагають обробки великої кількості даних на рівні застосунку. До атак на застосунок включають:

– SQL-ін'єкції – в цьому виді атаки зловмисники вставляють зловмисний SQL-код в форми або поля на веб-сайті з метою отримати несанкціонований доступ до бази даних та витягнути конфіденційну інформацію;

– XSS (Cross-Site Scripting) атаки – атаки, що включають в себе вставку зловмисного JavaScript-коду в веб-сторінки або видачу відповідей сервера зі зловмисним кодом, що виконується на браузері користувача, що може призвести до викрадення сесійних файлів або інших конфіденційних даних;

– CSRF (Cross-Site Request Forgery) атаки – ця атака вимагає використання відкритого сеансу користувача для надсилання несанкціонованих запитів на інші веб-сайти або дії в ім'я аутентифікованого користувача;

– Brute Force атаки – в цьому виді атаки зловмисники можуть намагатися зламати паролі або ідентифікатори входу, надсилаючи багато спроб авторизації за один раз;

– Phishing атаки – атаки, що включають в себе створення підроблених веб-сайтів або електронних листів, щоб викликати у користувачів введення конфіденційної інформації, такої як паролі або номери кредитних карт;

– Атаки на сеанси (Session Hijacking) – в цьому виді атаки зловмисники можуть намагатися вкрасти або використовувати активні сесії авторизації користувачів для несанкціонованого доступу;

– Атаки на багатофакторну аутентифікацію – в цьому виді атаки зловмисники можуть намагатися обійти механізми багатофакторної аутентифікації, такі як OTP (одноразовий пароль) або біометричні дані.

Атака на пропускну здатність [10] – це форма розподіленої атаки на відмову в обслуговуванні, за якої зловмисники намагаються заборонити доступ до ресурсу шляхом виснаження пропускної здатності мережі, заволодіваючи всім доступним обсягом смуги пропускання.

DDoS-атака на пропускну здатність, також відома як атака на мережеву пропускну здатність, спрямована на перевантаження і споживання всієї доступної пропускної здатності мережі або інтернет-каналу, що призводить до тимчасової відмови в обслуговуванні додатка або веб-сервісу. Ця атака не специфічно вражає додаток чи службу, але, навіть не завдаючи прямої шкоди, робить їх недоступними для користувачів через перевантаження мережевих ресурсів. До атак на пропускну здатність включають:

– TCP/IP флуд – атака, при якій зловмисник створює велику кількість TCP або IP пакетів, щоб перевантажити ресурси мережі або сервера, призводячи до перевищення пропускної здатності;

– UDP флуд – атака, яка використовує багато UDP пакетів, щоб забити пропускну здатність мережі або сервера;

– ICMP флуд – атака, при якій зловмисники використовують ICMP пакети для переповнення мережевого пристрою або сервера;

– HTTP флуд – атака, при якій зловмисник намагається перевантажити веб-сервер запитами HTTP;

– HTTPS флуд – атака, при якій зловмисник використовує шифровані HTTPS-запити для атаки веб-сервера;

– DNS флуд – атака, при якій зловмисники намагаються перевантажити DNS-сервери запитами на розрізнювання імен доменів;

– NTP ампліфікація – атака, при якій зловмисники використовують NTP-сервери (для відправки запитів із сфальсифікованими даними, що відправляються жертві та призводять до перевантаження її мережі.

У випадку атак на протоколи, атакують служби або протоколи, використовуючи вразливості в їх роботі. Наприклад, атаки на протокол DNS можуть призвести до відмови в обслуговуванні для веб-сайтів.

DDoS-атака на протоколи [11] – це специфічний вид атаки, спрямований на використання вразливостей або обмежень в мережевих або комунікаційних протоколах для зруйнування або недоступності цільового сервера чи мережі. Ця атака не обов'язково передбачає надсилання великої кількості трафіку, як у класичних DDoS-атаках, але може використовувати неправильні або спеціально сформовані пакети для спричинення відмови в обслуговуванні. До атак на протоколи включають:

– Атака перехоплення (Interception Attack) – під час такої атаки зловмисник спробує перехопити передачу даних між двома пунктами обміну інформацією, що в результаті може стати можливим через недостатнє шифрування або використання не захищених з'єднань;

– Атака повторного використання (Replay Attack) – атака, за якої зловмисник записує певний обмін даними і використовує цю інформацію пізніше для вигоди, наприклад, зловмисник може повторити попередню авторизацію для доступу до системи:

– Атака заміни (Man-in-the-Middle Attack, MITM) – під час даної атаки зловмисник вставляється між двома спільниками і перехоплює та може модифікувати обмін даними між ними, що в результаті дає зловмиснику можливість вивести відомості або здійснити інші шкідливі дії;

– Атака на переповнення буфера (Buffer Overflow Attack) – в цьому виді атаки зловмисник спробує ввести більше даних в буфер, ніж це передбачено протоколом, що в результаті може призвести до переповнення буфера і виконання шкідливого коду;

– Синтаксична атака (Syntax Attack) – атака, за якої зловмисник намагається вивести інші дії, змінивши синтаксис або структуру передачі даних, яка передбачена протоколом;

– Атака на перевірку справжності (Authentication Attack) – атака, за якої зловмисники можуть намагатися обійти або зламати механізми аутентифікації, які використовуються в протоколах для перевірки справжності користувачів.

Змішана DDoS-атака [12] – це складна та еволюційна форма атаки, яка поєднує в собі різні методи та види атак для досягнення більшої ефективності та обхід захисту цільового об'єкта. Ці атаки розробляються та виконуються зловмисниками з великими навичками та ресурсами, оскільки вони вимагають вдосконалених знань і координації.

1.3 Механізми реалізації DDoS-атак та методи захисту від них

DDoS-атаки можуть бути реалізовані різними механізмами, які атакувачі використовують для перевантаження цільового сервера або мережі. Розглянуті вище види DDoS-атак мають різні методи виконання та спрямованість, і захист від них вимагає вдосконалення мережевої безпеки та моніторингу, а також використання спеціалізованих рішень для виявлення і запобігання атакам. Нижче наведено опис механізмів реалізації різних видів DDoS-атак та методи захисту від них.

Атака на основі великого обсягу трафіку [13] проводиться в декілька етапів:

1. Збір ботнету. Зловмисники зазвичай створюють ботнет, що складається з великої кількості комп'ютерів або інших пристроїв, які можуть бути заражені шкідливими програмами, такими як віруси, троянці, чи черви. Ці компрометовані пристрої стають частиною ботнету, і їхні власники зазвичай не мають уявлення про їхню участь у атаках.

2. Віддалене управління. Зловмисники використовують віддалене управління для керування ботнетом і відправки команд для запуску DDoS-атак. Вони можуть визначати, коли розпочати атаку та які цільові сервери або мережі атакувати.

3. Спам і фальшиві запити. Коли настане момент атаки, зловмисники відправляють велику кількість трафіку до цільового сервера чи мережі. Це може виглядати як HTTP-запити, UDP або TCP-пакети, а також інші мережеві протоколи. Важливо відзначити, що цей трафік зазвичай носить характер

"неправильного" або випадкового трафіку, і він може бути схожий на справжній, але надзвичайно інтенсивний.

4. Ампліфікація. У деяких випадках зловмисники використовують ампліфікацію, щоб підсилити обсяг атаки. Вони можуть відправляти запити до серверів, які відомі своєю здатністю створювати велику кількість відповідей на кожний запит. Наприклад, атака ампліфікації може використовувати DNS-сервери, щоб відправити невеликі запити, але призводити до значної кількості великих DNS-відповідей, що спричиняє збільшення обсягу трафіку.

5. Фальшиві IP-адреси. Зловмисники можуть також використовувати фальшиві IP-адреси, щоб приховати свою ідентичність і зробити атаку більш ефективною.

6. Видалення захисту. Деякі атаки можуть бути спрямовані на обхід захисту, який встановлений на цільовому сервері, шляхом використання різних технік або здійснення подібних атак одночасно, щоб розірвати обрані об'єкти.

7. Закриття доступу. Метою атаки є перевантаження цільового об'єкта до такої міри, що він стає недоступним для легітимних користувачів, які намагаються отримати доступ до ресурсу.

Отже, основна ідея в атаках на основі великого обсягу трафіку – це перевантаження цільового сервера або мережі настільки, щоб вони стали недоступними. Для захисту від таких атак важливо мати системи фільтрації, аналізу трафіку та виявлення аномалій, які можуть реагувати на атаки та зменшити їхній вплив.

Атака на ресурси [14] здійснюється в декілька етапів:

1. Підготовка і збір ботнету. Як і в інших видах DDoS-атак, зловмисники створюють ботнет, що складається з компрометованих пристроїв. Ці пристрої можуть бути заражені вірусами, троянцями, чи червами, і зазвичай знаходяться в різних кутках Інтернету. Зловмисники використовують цей ботнет для запуску атаки.

2. Надсилання запитів на ресурси. Після створення ботнету зловмисники надсилають велику кількість запитів до цільового сервера чи мережі. Ці запити

зазвичай вимагають великої обчислювальної потужності чи інших ресурсів для обробки.

3. Використання вразливостей або надмірний трафік. Зловмисники можуть використовувати вразливості в програмному забезпеченні цільового сервера або мережі, або ж просто надсилати запити на такий обсяг трафіку, який перевантажує ресурси, що не можуть бути оброблені.

4. Вичерпання ресурсів. Основною метою атаки на ресурси є вичерпання обчислювальних, пам'яті, мережевих, або інших ресурсів цільового сервера або мережі. Це може призвести до зниження продуктивності або навіть до відмови в обслуговуванні.

Отже, для захисту від DDoS-атак на ресурси важливо використовувати різноманітні стратегії. Це включає в себе застосування систем фільтрації та виявлення аномалій для розпізнавання та блокування шкідливого трафіку, а також розробку плану реагування на інциденти для швидкого відновлення роботи системи в разі атаки.

Атака на застосунок [15] проводиться в декілька етапів:

1. Вибір цільового додатка або сервісу. Зловмисник вибирає конкретний додаток або сервіс, який він бажає атакувати. Це може бути веб-сайт, веб-додаток, поштовий сервер, онлайн-ігри тощо.

2. Збір інформації. Зловмисник проводить розвідку для збору інформації про цільовий об'єкт. Це включає в себе виявлення слабких місць, вимірювання ресурсів та аналіз структури додатка.

3. Створення ботнету. Зловмисник створює ботнет, що складається з кількох тисяч або навіть мільйонів комп'ютерів, які можуть бути збурені для атаки одночасно. Зазвичай для цього використовуються комп'ютери, заражені вірусами або іншими шкідливими програмами.

4. Запити на цільовий сервер. Запити на цільовий сервер або додаток посилаються великою кількістю ботнет-комп'ютерів. Ці запити можуть бути запитами HTTP, DNS-запитами або іншими типами запитів, які характерні для цільового додатка.

5. Перевантаження сервера. Велика кількість запитів перевантажує цільовий сервер або додаток. Сервери не можуть обробляти всі запити одночасно і починають сповільнюватися або відмовляти в обслуговуванні законних користувачів.

Отже, DDoS-атака на застосунок може призвести до тимчасової недоступності цільового додатка або сервісу для користувачів. Це може завдати значних фінансових збитків та пошкодити репутацію організації. Важливо враховувати, що атаки на застосунок вимагають багато ресурсів і координації з боку зловмисника, а також ефективного захисту з боку цільового оператора системи. Тому захист від таких атак передбачає використання технологій, які виявляють та нейтралізують атаки, а також можливість масштабувати інфраструктуру для витримки великого обсягу трафіку. Атакований оператор системи повинен вживати заходів для виявлення і фільтрації фіктивних запитів, які прийшли від ботнета. Це може включати в себе застосування різних методів фільтрації, розробку правил брандмауера, використання кешування та інші техніки для відсіювання трафіку.

Атака на пропускну здатність [16] здійснюється в декілька етапів:

1. Вибір цілі. Зловмисник обирає цільовий об'єкт, який може бути будь-яким сервером або мережевим вузлом, доступним через інтернет.

2. Створення ботнету. Зловмисник формує ботнет, що складається з багатьох комп'ютерів, які можуть бути заражені вірусами або включені в мережу без відома їхніх власників. Цей ботнет буде використовуватися для надсилання великої кількості трафіку до цільового об'єкта.

3. Запити до цілі. За допомогою ботнету зловмисник відправляє велику кількість запитів до цільового об'єкта. Це може бути ICMP (Ping) запитами, UDP або TCP пакетами, або іншими мережевими пакетами.

4. Засмічення пропускної здатності. Велика кількість трафіку, яку генерує ботнет, перевантажує мережевий канал, який з'єднує цільовий об'єкт з інтернетом. Як результат, цільовий об'єкт може втратити доступ до інтернету або

стати недоступним для користувачів через споживання всієї пропускної здатності.

Отже, атака на пропускну здатність може призвести до тимчасової недоступності цільового об'єкта та завдати шкоди бізнесу або інфраструктурі. Для попередження таких атак необхідно використовувати ефективний мережевий захист та розробити план відновлення після атаки для швидкого відновлення роботи мережі. Для захисту від таких атак оператори систем повинні використовувати мережеві пристрої, такі як файерволи та брандмауери, які можуть виявляти та фільтрувати шкідливий трафік. Також важливо використовувати інструменти моніторингу мережі для виявлення аномалій у трафіку.

Атака на протоколи [17] проводиться в декілька етапів:

1. Вибір цілі. Зловмисник вибирає цільовий сервер або мережевий вузол, на який буде спрямована атака.

2. Вибір протоколу. Зловмисник аналізує цільовий сервер, щоб знайти вразливості в мережевих або комунікаційних протоколах, які використовуються для з'єднання з сервером. Це може бути протоколи, такі як TCP, UDP, ICMP, HTTP, FTP тощо.

3. Зламані або фальшиві запити. Зловмисник надсилає серверу спеціально сформовані, неправильні або підроблені запити, які можуть викликати вразливості у цільовому протоколі або вичерпати ресурси на сервері. Наприклад, атаки можуть включати в себе SYN/ACK флуд (для атак на TCP), Ping of Death атаки (для атак на ICMP), або HTTP запроси з великою кількістю заголовків (для атак на HTTP).

4. Перевантаження або використання вразливості. Надмірне число запитів або експлуатація вразливостей у протоколах може призвести до перевантаження сервера або витоку ресурсів. Це може призвести до зниження продуктивності сервера або його відмови в обслуговуванні.

Отже, атаки на протоколи можуть бути складними для виявлення та лікування через їхню специфічність. Вони можуть призвести до недоступності

серверів та серйозної роботи бізнесу. Захист передбачає не тільки фільтрацію некоректного трафіку, але й вирішення вразливостей у протоколах та розробку стратегій відновлення після атаки для мінімізації завданих шкод. Оператори мереж повинні використовувати різноманітні механізми захисту, такі як файерволи, IPS (Intrusion Prevention Systems), WAF (Web Application Firewalls) та системи моніторингу мережі, щоб виявити та блокувати аномальний трафік, пов'язаний із спробами атаки на протоколи.

Змішані атаки [18] здійснюється в декілька етапів:

1. Вибір цілі. Зловмисник обирає цільовий об'єкт, який може бути веб-сайтом, онлайн-сервісом, мережею або іншими ресурсами.

2. Сканування та розвідка. Зловмисник проводить розвідку для визначення слабких місць та потенційних вразливостей цільового об'єкта. Він може використовувати зовнішні інструменти або власні сканери для цього.

3. Соціальний інжиніринг. Зловмисник може використовувати соціальний інжиніринг або фішинг, щоб отримати доступ до логінів, паролів або іншої конфіденційної інформації. Це може включати в себе відправку шкідливих листів або маніпулювання користувачами.

4. Атака на протоколи. Зловмисник може використовувати атаки на протоколи (наприклад, SYN/ACK флуд або ICMP флуд) для перевантаження мережевого ресурсу цільового об'єкта. Це може бути поєднано з атаками на додаток для підсилення ефекту.

5. Атака на застосунок. Зловмисник може використовувати атаки на застосунок (наприклад, HTTP або HTTPS флуд, або SQL ін'єкція) для надсилання запитів, які спрямовані на вразливості програмного забезпечення цільового додатка.

6. Використання ампліфікації. Зловмисник може використовувати ампліфікацію, де він надсилає невеликі запити до засобів ампліфікації (наприклад, отримання великих відповідей від DNS-серверів) для підсилення атаки.

Отже, змішані DDoS-атаки є особливо небезпечними через їхню складність та здатність обходити типові методи захисту. Підтримка регулярного моніторингу та посилення захисту є ключовими аспектами для захисту від таких атак. Захист від змішаних DDoS-атак вимагає розробки комплексних стратегій захисту, які включають в себе виявлення аномального трафіку, моніторинг мережі, застосування файерволів, брандмауерів, системи WAF та системи інтелектуального аналізу трафіку. Після атаки також необхідно мати план відновлення, який включає в себе аналіз і підсилення захисту та відновлення роботи ресурсу.

1.4 Вплив DDoS-атак на веб-ресурси

DDoS-атаки можуть мати серйозний і негативний вплив на веб-ресурси. До наслідків DDoS-атак на веб-сайти та інші веб-ресурси, що мають негативний вплив, відносять:

1. Недоступність сервісу (Service Unavailability).
2. Збитки від втрати доступу (Loss of Revenue).
3. Порушення репутації (Reputation Damage).
4. Витрати на відновлення (Recovery Costs).
5. Втрати клієнтів та користувачів (Loss of Customers/Users).
6. Витрати на захист (Protection Costs).
7. Перспективність майбутніх атак (Risk of Future Attacks).

Недоступність сервісу [19] – це один з найочевидніших та найпоширеніших наслідків DDoS-атаки для веб-ресурсів. Це являє собою неможливість користувачів отримати доступ до веб-сайту чи онлайн-сервісу через перевантаження серверів та мережі атаками. З даного наслідку випливають:

– перевантаження сервера і мережі – DDoS-атаки надсилають велику кількість запитів або трафіку на цільовий сервер чи мережу, оскільки сервер чи

мережа мають обмежені ресурси (процесори, пам'ять, пропускна спроможність), ця надмірна навантаженість може спричинити їх перевантаження;

– спад швидкості відповіді – перевантажений сервер стає неспроможним швидко обробляти запити користувачів, як наслідок, веб-сайт може відкриватися з великими затримками або взагалі не відкриватися, а користувачі можуть очікувати дуже повільного завантаження сторінок або взагалі не отримувати відповіді;

– відмова в обслуговуванні – DDoS-атаки в сутності є спробою заповнити всі ресурси сервера або мережі настільки, щоб вони не могли більше обслуговувати легітимні запити, що призводить до відмови в обслуговуванні для легітимних користувачів, оскільки їхні запити не можуть бути оброблені;

– втрати прибутку – веб-сайти, особливо комерційні, можуть втрачати прибуток через недоступність, оскільки користувачі, які не можуть отримати доступ до продуктів або послуг, можуть звертатися до конкурентів, що може призвести до значних втрат фінансових ресурсів;

– вплив на користувачів і клієнтів – недоступність сервісу може призвести до розлючення користувачів та клієнтів, оскільки вони можуть вважати, що організація не здатна забезпечити надійний сервіс, і це може вплинути на їхню довіру та лояльність;

– затримки у вирішенні проблеми – після завершення DDoS-атаки може знадобитися час для відновлення нормального функціонування веб-сайту чи сервісу, що означає, що користувачі можуть продовжувати стикатися з недоліками в роботі протягом певного часу;

– вплив на репутацію – недоступність сервісу може вплинути на репутацію компанії або бренду, оскільки користувачі можуть сприймати це як недбалість або нездатність організації забезпечити стабільну роботу своїх ресурсів;

Недоступність сервісу є одним із найбільш очевидних та завдавальних наслідків DDoS-атак, і вона може мати серйозний вплив на бізнес та репутацію

організації. Тому захист від DDoS-атак та плани відновлення стають надзвичайно важливими для будь-якої онлайн-організації.

Збитки від втрати доступу [20] – відображає фінансові втрати, які компанії та організації можуть понести через недоступність своїх онлайн-сервісів. З даного наслідку випливають:

– втрата прибутку – є найочевиднішим ефектом DDoS-атаки в цьому контексті, оскільки веб-сайти та інші веб-ресурси часто генерують дохід через продаж товарів і послуг, рекламні партнерства або підписки, тому якщо користувачі не можуть отримати доступ до сайту чи сервісу через атаку, це може призвести до втрати потенційних продажів та доходу;

– зменшення конверсії – під впливом DDoS-атак, користувачі можуть втратити зацікавленість та відвідати конкурентів, що особливо негативно вплине на онлайн-бізнеси, які залежать від конверсії, наприклад, інтернет-магазини, де збитки від втрати конверсії можуть бути значними;

– втрати рекламного доходу – якщо веб-сайт має рекламу, то він може отримувати дохід від рекламодавців, тому недоступність сайту може призвести до втрати рекламного доходу, оскільки рекламодавці можуть не бути задоволені низьким обсягом переглядів та кліків;

– зниження лояльності користувачів – користувачі, які стикаються з постійною недоступністю веб-сайту або сервісу через DDoS-атаки, можуть втратити зацікавленість у компанії та перейти до інших, що може призвести до втрати лояльних клієнтів на довгий термін;

– психологічні втрати - поза фінансовими аспектами, DDoS-атаки можуть викликати стрес та негативний психологічний вплив на команду та керівництво організації, в результаті вони можуть відчувати безпорадність перед атакою та невпевненість у можливості захистити ресурси;

– потреба у фінансових ресурсах – після DDoS-атаки організація може витрачати фінансові ресурси на відновлення та зміцнення захисту, а витрати на попередження майбутніх атак та відновлення інфраструктури можуть бути значними.

Збитки від втрати доступу є серйозною загрозою для веб-ресурсів і бізнесу загалом. Це може вплинути на фінансовий стан організації та її репутацію. Тому важливо мати добре розроблені плани відновлення та захисту від DDoS-атак.

Порушення репутації [20] – відображає вплив атаки на відносини з користувачами, клієнтами, партнерами та громадськістю, що може мати довгостроковий ефект на бізнес. З даного наслідку випливають:

– втрата довіри користувачів – коли веб-ресурс стає недоступним через DDoS-атаку, користувачі можуть почувати себе розчарованими та незадоволеними, в результаті вони можуть втратити довіру до організації, вважаючи, що та не може забезпечити надійний та доступний сервіс;

– негативні відгуки та рейтинги – користувачі, які стикаються з недоступністю веб-сайту або сервісу через DDoS-атаку, можуть залишати негативні відгуки та оцінки на різних форумах, соціальних медіа і в інших відкритих джерелах, що може погіршити репутацію організації;

– вплив на бренд – порушення репутації може мати великий вплив на бренд організації, а негативні новини про недоступність веб-сайту або сервісу можуть поширюватися швидко та призводити до падіння довіри до бренду;

– потенційні втрати клієнтів і клієнтських відносин - порушення репутації може призвести до втрати клієнтів та клієнтських відносин, оскільки користувачі, які втратили довіру до організації через недоступність, можуть вибрати конкурентів, що може призвести до зменшення прибутку та ринкової частки;

– вплив на інвесторів та партнерів – порушення репутації може також вплинути на стосунки з інвесторами та партнерами, як наслідок, інвестори можуть вагатися чи інвестувати в компанію з поганою репутацією, а партнери можуть шукати інші співпраці;

– психологічний вплив на команду та керівництво – для команди та керівництва організації DDoS-атака може стати психологічною навантаженням, тому вони можуть відчувати стрес та безпорадність перед атакою та невпевненість у здатності захистити ресурси;

– вплив на прийняття рішень – порушення репутації може вплинути на прийняття рішень в організації, в результаті керівництво може відчувати тиск від громадськості, акціонерів та інших стейкхолдерів щодо прийняття конкретних заходів для відновлення репутації.

Вплив DDoS-атак на репутацію може бути дуже серйозним та довгостроковим. Позбавлення від негативного впливу на репутацію може вимагати великих зусиль від організації та вдосконалення стратегій безпеки.

Витрати на відновлення [21] охоплюють витрати, які організація повинна понести після завершення DDoS-атаки для відновлення нормального функціонування своїх ресурсів та забезпечення їхньої безпеки. З даного наслідку випливають наступні витрати на відновлення:

– аналіз і розслідування атаки – по завершенню DDoS-атаки організація повинна провести ретельний аналіз та розслідування атаки, щоб з'ясувати її джерело, методи та мету, що може вимагати залучення фахівців з кібербезпеки та інженерів;

– відновлення інфраструктури – перевантажений сервер або мережа може потребувати часу та ресурсів для відновлення нормальної роботи, в результаті витрати можуть включати в себе відновлення серверів, мережевого обладнання та програмного забезпечення, яке може бути пошкодженим або вразливим;

– заходи безпеки та захисту – після DDoS-атаки організація може вживати заходи для зміцнення безпеки та запобігання майбутнім атакам, що може включати в себе встановлення файерволів, брандмауерів, ідентифікацію та запобігання вразливостям;

– заміна атакованих апаратних засобів – якщо атака призвела до фізичного пошкодження апаратних засобів, то їх можливо буде потрібно замінити або відновити, що може бути вкрай дорогою процедурою;

– заходи моніторингу та виявлення - організація може вкласти кошти у системи моніторингу та виявлення, щоб реагувати на атаки на ранніх стадіях і запобігати їхньому наступному розвитку;

– співпраця з постачальниками послуг – в деяких випадках організація може оплачувати додаткові послуги від свого хостинг-постачальника або постачальника CDN (Content Delivery Network), щоб допомогти захистити свої ресурси від DDoS-атак;

– заходи для відновлення довіри користувачів – після атаки організація може проводити комунікаційні заходи для відновлення довіри користувачів та інших стейкхолдерів, що може включати в себе публічні оголошення, заяви про безпеку та рекламні заходи;

– витрати на підготовку до майбутніх атак – організація може інвестувати в підготовку та навчання свого персоналу для впровадження кращих практик щодо кібербезпеки та впровадження заходів безпеки.

Витрати на відновлення після DDoS-атаки можуть бути значними і можуть тривати впродовж тривалого періоду. До цього важливо додати витрати на захист від майбутніх атак. Організації повинні бути готові до можливих наслідків DDoS-атак та мати плани відновлення та захисту для забезпечення стійкості своїх веб-ресурсів.

Втрати клієнтів та користувачів [20] – відображає втрату активних користувачів та клієнтів через недоступність сайту або сервісу під час атаки. З даного наслідку випливають:

– втрати прибутку – коли користувачі не можуть отримати доступ до веб-сайту чи сервісу через DDoS-атаку, організація втрачає прибуток, який вони могли б заробити від цих користувачів, що особливо важливо для бізнесів, які залежать від постійного потоку клієнтів;

– зменшення конверсії – DDoS-атаки можуть призвести до зменшення конверсії, оскільки користувачі, які не можуть отримати доступ до сайту, не можуть виконувати дії, які призводили б до покупок чи інших конверсійних подій;

– втрати лояльних клієнтів – користувачі, які регулярно використовують веб-ресурс, можуть втратити лояльність, якщо вони стикаються з постійною

недоступністю, в результаті вони можуть перейти до конкурентів, які пропонують стабільніший сервіс;

– негативний вплив на репутацію – недоступність сайту або сервісу може вплинути на репутацію організації, а користувачі можуть сприймати це як недбалість або неспроможність організації забезпечити надійний сервіс;

– втрати на довгостроковому рівні – надзвичайно важливо розуміти, що втрати клієнтів та користувачів можуть відчуватися не тільки в момент DDoS-атаки, але і на довгостроковому рівні, тому користувачі можуть зберігати негативне враження і уникати використання веб-ресурсу в майбутньому;

– витрати на привернення нових користувачів – втрата клієнтів може змусити організацію вкладати додаткові кошти в маркетинг та рекламні кампанії для привернення нових користувачів та відновлення свого клієнтського бази;

– зміна інтересів користувачів – користувачі, які стикаються з недоступністю веб-ресурсу, можуть змінити свої інтереси та поведінку, тому вони можуть шукати альтернативні ресурси або послуги, які вони раніше не розглядали.

Втрати клієнтів та користувачів можуть бути серйозними і довгостроковими наслідками DDoS-атак. Організації повинні розглядати не тільки негайні втрати, але і потенційний вплив на свою репутацію та відносини з клієнтами. Для зменшення цього впливу важливо мати міри безпеки та плани відновлення в місці.

Витрати на захист [22] – відображає витрати, які організація має здійснити для запобігання майбутнім DDoS-атакам і забезпечення безпеки своїх веб-ресурсів. З даного наслідку випливають наступні витрати на захист:

– закупівля анти-DDoS рішень – одна з основних витрат, яка може включати в себе закупівлю і встановлення апаратного або програмного обладнання, яке спеціалізується на виявленні та блокуванні DDoS-атак;

– підписка на хмарні анти-DDoS служби – деякі організації можуть вирішити використовувати послуги хмарних анти-DDoS служб, що передбачає

плату за підписку на послуги, які надаються спеціалізованими службами для виявлення та фільтрації DDoS-трафіку перед тим, як він досягає веб-ресурсу;

– найм фахівців з кібербезпеки – організації можуть наймати фахівців з кібербезпеки для посилення свого командного складу та моніторингу заходів безпеки, а витрати на заробітну плату та навчання фахівців можуть бути значними;

– постійне оновлення заходів безпеки – захист від DDoS-атак вимагає постійного оновлення технічних та організаційних заходів безпеки, що включає в себе регулярне оновлення програмного забезпечення та апаратного забезпечення, а також аналіз та підвищення рівня безпеки;

– проведення тренувань та навчання персоналу – організації повинні інвестувати в тренування свого персоналу щодо розпізнавання та відповіді на DDoS-атаки, що в результаті допомагає підготувати команду до ефективної реакції на інциденти безпеки;

– використання захисту на рівні мережі та додатків – витрати також можуть включати в себе реалізацію захисту на рівні мережі та додатків, таких як файерволи, брандмауери та системи виявлення вторгнень (IDS/IPS);

– моніторинг та аналіз – організації також повинні витрачати кошти на системи моніторингу та аналізу, щоб вчасно виявляти аномальний трафік та потенційні атаки;

– плани відновлення і реагування – важливо розробляти та впроваджувати плани відновлення та реагування на DDoS-атаки, що включає в себе створення процедур та документації для відновлення сервісу після атаки;

– заходи для мінімізації впливу на користувачів - витрати також можуть бути пов'язані з розробленням заходів для мінімізації впливу DDoS-атак на користувачів, таких як перенаправлення трафіку або надання інформації про тимчасову недоступність.

Витрати на захист від DDoS-атак можуть бути значними, але це важливий інвестиційний крок для забезпечення безпеки веб-ресурсів та надійності

послуги. Недооцінка цих витрат може призвести до серйозних наслідків в разі атаки.

Перспективність майбутніх атак [22] – відображає можливість подальших DDoS-атак та їхніх наслідків для безпеки та стабільності веб-ресурсів. З даного наслідку випливають:

– повторні атаки – після першої DDoS-атаки організація може стати об'єктом повторних атак, оскільки зловмисники можуть намагатися використовувати ті ж самі або нові методи атаки для завдання шкоди веб-ресурсу;

– збільшення інтенсивності атак – зловмисники можуть збільшити інтенсивність DDoS-атак з часом, використовуючи більше ботнетів або більше заражених пристроїв, що може призвести до більших збитків та недоступності веб-ресурсу;

– використання нових методів – з часом зловмисники можуть розвивати та вдосконалювати методи DDoS-атак, наприклад, вони можуть використовувати нові види атак, які важко виявляти та блокувати;

– зміна цілей атак – зловмисники можуть змінювати цілі своїх атак для виконання конкретних цілей або завдання шкоди певним ресурсам організації, що може включати в себе атаки на додатки, бази даних або інші важливі компоненти інфраструктури;

– економічні збитки – перспективність майбутніх атак також враховує економічні збитки, пов'язані з необхідністю інвестувати в заходи безпеки для мінімізації ризику атак та відновлення в разі їхнього успішного проведення;

– зміна обставин – зміни в інфраструктурі, бізнес-моделі або технологічному середовищі організації можуть збільшитися ризики майбутніх DDoS-атак, наприклад, розширення мережі або впровадження нових сервісів може зробити веб-ресурси більш вразливі від атак;

– вплив на бренд та репутацію – майбутні DDoS-атаки можуть значно позначитися на бренді та репутації організації, що може призвести до втрат клієнтів та клієнтських відносин.

Для управління ризиками майбутніх DDoS-атак організаціям необхідно вдосконалювати свої заходи безпеки, реалізувати системи моніторингу та виявлення, тримати оновлені резервні копії даних і створювати плани відновлення. Крім того, важливо бути готовим до можливих атак і реагувати на них швидко і ефективно.

1.5 Висновки до розділу 1

Перший розділ присвячено дослідженню теоретичних аспектів DDoS-атак. В результаті проведеного дослідження:

1. Було проаналізовано нормативно-правову базу щодо забезпечення національної безпеки України в інформаційній сфері від DDoS-атак, яка включає: норми Конституції України; Указ Президента України «Про Стратегію національної безпеки України»; Положення «Про організаційно-технічну модель кіберзахисту»; «Загальні вимоги до кіберзахисту об’єктів критичної інфраструктури»; Закон України «Про основні засади забезпечення кібербезпеки України», тощо. Даний аналіз дозволяє нам зробити висновок про наявність достатньої нормативно-правової бази щодо перспектив захисту національної безпеки України, зокрема в інформаційній сфері, від DDoS-атак.

2. Було з’ясовано поняття DDoS-атаки, що являє собою форму кібератаки, яка полягає в спробі перевантажити або знеструмити цільовий сервер або мережу, надсилаючи велику кількість запитів або трафіку. Також було досліджено види DDoS-атак, а саме: атака на основі великого обсягу трафіку; атака на ресурси; атака на застосунок; атака на пропускну здатність; атака на протоколи; змішані атаки. Це дозволяє нам краще розуміти механізми та можливі наслідки різних типів DDoS-атак, що є важливим для розробки ефективних систем захисту від цих загроз у кіберпросторі.

3. Було визначено механізми реалізації різних видів DDoS-атак та з’ясовано методи захисту від них. Це допомогло зрозуміти, як розвиватиметься

подальший процес розробки програмного продукту в напрямку захисту від атак типу DDoS.

4. Було визначено вплив DDoS-атак на веб-ресурси, а саме негативний вплив. До наслідків DDoS-атак на веб-сайти та інші веб-ресурси, що мають негативний вплив, відносять: недоступність сервісу; збитки від втрати доступу; порушення репутації; витрати на відновлення; втрати клієнтів та користувачів; витрати на захист; перспективність майбутніх атак. Це слугувало для розуміння, що важливо врахувати при розробці системи захисту від DDoS-атак.

РОЗДІЛ 2.
ІНСТРУМЕНТИ І ТЕХНОЛОГІЇ РОЗРОБКИ СИСТЕМИ ДЛЯ ПРОТИДІЇ DDOS-АТАК

2.1 Вибір мови програмування для розробки системи захисту від DDoS-атак

При проектуванні та розробці системи захисту від DDoS-атак вибір мови програмування стає стратегічно важливим етапом, оскільки від цього вибору залежить ефективність системи та швидкість реакції на потенційні загрози. Мова програмування повинна бути здатна обробляти великі обсяги даних за короткий період часу, а також повинна бути ефективною для виявлення та реагування на DDoS-атаки.

У ході аналізу вибору мови програмування, ми виявили, що Python є оптимальним вибором для розробки систем захисту від DDoS-атак з кількох причин [23]:

1. Ефективність обробки даних. Однією з ключових вимог до мови програмування є її здатність ефективно обробляти великі обсяги даних з високою продуктивністю. Python відзначається високою продуктивністю, особливо при використанні оптимізованих алгоритмів обробки даних. Це важливо для систем захисту від DDoS-атак, де швидкість аналізу та реакції грає критичну роль у протидії атакам.

2. Швидкість навчання та розробки. Простота та легкість вивчення Python роблять його ідеальним вибором для розробників, які мають різний рівень досвіду та хочуть швидко освоїти новий інструмент. Швидкість розробки є критичною для систем захисту, оскільки це дозволяє швидко впроваджувати та тестувати нові функції чи алгоритми реакції на DDoS-атаки.

3. Багата екосистема бібліотек і фреймворків. Python славиться розгалуженою екосистемою бібліотек та фреймворків. У контексті систем

захисту від DDoS, це означає, що розробники мають доступ до різноманітних інструментів для вирішення конкретних викликів. Наявність готових рішень полегшує і прискорює розробку системи.

4. Можливість масштабування. Умови змінюються, обсяги даних можуть різко збільшуватися, і система захисту повинна бути здатною адаптуватися до нових умов. Python відзначається легкістю масштабування, що робить його відмінним вибором для систем, які повинні швидко реагувати на зростання обсягів даних та навантаження.

Узагальнюючи, вибір Python для системи захисту від DDoS-атак обґрунтовується його здатністю ефективно обробляти дані, швидкістю навчання та розробки, розгалуженою екосистемою, а також легкістю масштабування. Ці аспекти створюють надійну основу для впровадження ефективної та відзначеної системи захисту від DDoS-атак.

До того ж, ми з’ясували, що вибір мови програмування Python для розробки системи захисту від DDoS-атак, має кілька конкретних переваг, а саме [24]:

1. Обробка потоків даних. Python володіє потужними інструментами для обробки потоків даних, що робить його ідеальним вибором для систем захисту від DDoS-атак, де необхідно аналізувати великі обсяги мережевого трафіку в реальному часі. Вбудовані функції Python сприяють швидкій та ефективній обробці потоків даних, що є важливим для вчасного виявлення та реагування на аномалії.

2. Виявлення аномалій. Python дозволяє використовувати різноманітні техніки, включаючи статистичний аналіз та методи машинного навчання, для виявлення аномалій у мережевому трафіку. Висока гнучкість та розширена функціональність мови сприяють створенню надійної системи виявлення ненормальних патернів.

3. Реагування на атаки. Python дозволяє ефективно реагувати на DDoS-атаки. За допомогою вбудованих інструментів можна автоматично блокувати IP-адреси, змінювати налаштування мережі та вживати інші заходи для захисту системи. Це забезпечує надійний механізм протидії потенційним загрозам.

В той же час, ми виявили і декілька недоліків мови програмування Python для розробки системи захисту від DDoS-атак, а саме [25]:

1. Швидкодія. Однією з основних критичних зауважень стосовно Python є його швидкодія в порівнянні з іншими мовами програмування, такими як C++ чи Java. Для деяких завдань захисту від DDoS-атак, де швидкодія є критичною, це може бути певним обмеженням.

2. Оптимізація пам'яті. Python може використовувати більше пам'яті порівняно з деякими іншими мовами програмування. Це може впливати на продуктивність системи в умовах обробки великих обсягів даних. Потрібно ретельно оптимізувати роботу з пам'яттю для забезпечення ефективності системи.

Для нашого переконання, що мова програмування Python для нашої майбутньої програми є оптимальним варіантом, ми здійснили порівняння мови програмування Python з іншими мовами програмування.

Порівнюючи мову програмування Python з іншими мовами програмування, такими як Java, C++ чи Go, важливо враховувати конкретні потреби майбутньої програми. Кожна мова має свої унікальні особливості, які можуть вплинути на стратегічне рішення при розробці системи захисту від DDoS-атак.

Порівнюючи мови програмування Python та Java, ми виявили переваги, як в одної, так і в іншої мови програмування, що в результаті є важливим аспектом для проектування та розробки майбутньої програми.

Переваги Python:

1. Простота та швидкість навчання. Python славиться своєю простотою та легкістю вивчення, що полегшує швидке освоєння нових розробників.

2. Ефективність обробки даних. Висока продуктивність Python при оптимізованих алгоритмах робить його відмінним вибором для обробки великих обсягів даних.

3. Широкий вибір бібліотек. Розгалужена екосистема бібліотек робить Python гнучким для різних завдань.

Переваги Java [26]:

1. Переносимість та висока швидкість. Java відома своєю переносимістю завдяки використанню JVM, але в той же час може мати вищу швидкість порівняно із Python.

2. Велика бібліотека. Java пропонує обширну бібліотеку, що полегшує розробку різних застосунків.

Порівнюючи мови програмування Python та C++, ми виявили переваги, як в одної, так і в іншої мови програмування, що в результаті є важливим аспектом для проектування та розробки майбутньої програми.

Переваги Python:

1. Простота та швидкість розробки. Python дозволяє розробникам швидко втілювати ідеї завдяки простоті синтаксису.

2. Масштабованість. Python легко масштабується, але може мати меншу швидкодію порівняно з C++.

Переваги C++ [27]:

1. Швидкість та продуктивність. C++ відомий своєю високою швидкістю виконання завдяки компіляції в машинний код.

2. Глибокий рівень системного контролю. C++ надає більше контролю над системою, але вимагає від розробників більшого досвіду.

Порівнюючи мови програмування Python та Go, ми виявили переваги, як в одної, так і в іншої мови програмування, що в результаті є важливим аспектом для проектування та розробки майбутньої програми.

Переваги Python:

1. Швидкість розробки. Python виграє в простоті та швидкості розробки.

2. Висока продуктивність. Використання багатьох бібліотек Python сприяє високій продуктивності.

Переваги Go [28]:

1. Швидкодія та конкурентність. Go відома своєю швидкодією та здатністю ефективно працювати в конкурентних умовах..

2. Низьке використання пам'яті. Go оптимізована для ефективного використання пам'яті.

В результаті аналізу порівняння мов програмування, ми з’ясували, що такі мови програмування, як Java, C++, Go, не є оптимальним вибором для проектування та розробки майбутньої програми через деякі їх недоліки:

1. Java і C++ можуть вимагати більше коду для вираження тих самих концепцій порівняно з Python, що в результаті може збільшити обсяг коду та час розробки.

2. Java і Go є менш гнучкими щодо деяких аспектів програмування порівняно з Python, що може ускладнити роботу з деякими завданнями.

3. C++ відомий своєю високою швидкодією, але вимагає більшого рівня досвіду.

4. C++ і Go можуть мати меншу кількість готових бібліотек порівняно з Python, що в результаті може вимагати додаткових зусиль у розробці та відладці.

Отже, для проектування та розробки майбутньої програми нами було обрано мову програмування Python через поєднання простоти навчання, швидкості розробки та широкого спектру бібліотек і фреймворків. Для систем захисту від DDoS-атак, де важливо швидко реагувати на нові загрози та підтримувати високий рівень ефективності, Python виявляється оптимальним вибором, не зважаючи на деякі обмеження у швидкодії та оптимізації пам'яті.

Також, під час дослідження, ми з’ясували кілька конкретних прикладів того, як Python можна використовувати для розробки систем захисту від DDoS-атак [29]:

1. Система виявлення аномальних потоків. Система може використовувати Python для аналізу даних мережевого трафіку за допомогою статистичного аналізу або машинного навчання. Це може допомогти виявити ненормальний трафік, який може бути ознакою DDoS-атаки.

2. Система блокування IP-адрес. Система може використовувати Python для блокування IP-адрес, які надсилають аномальний трафік. Це може допомогти захистити систему від DDoS-атаки.

3. Система розподілу навантаження. Система може використовувати Python для розподілу трафіку між декількома серверами. Це може допомогти захистити систему від DDoS-атаки, розподіляючи навантаження між серверами.

В результаті аналізу та досліджень, ми з’ясували, що Python є потужним інструментом, який можна використовувати для розробки ефективних систем захисту від DDoS-атак.

2.2 Використання середовища розробки PyCharm

Однією з ключових складових успішної розробки систем захисту від DDoS-атак є вибір відповідного середовища розробки (IDE – Integrated Development Environment), яке сприяє ефективності, зручності та швидкості процесу створення програмного забезпечення.

Для розробки системи захисту від DDoS-атак з обраною мовою програмування Python, необхідно використати середовище розробки PyCharm, яке є інтегрованим середовищем розробки (IDE) для Python, що дозволяє розробникам створювати, тестувати та запускати програми Python.

Під час проведеного дослідження, ми з’ясували, що PyCharm має ряд функцій, які роблять його зручним для розробки систем захисту від DDoS-атак, а саме [30]:

1. Вбудований відладчик – є невід'ємною частиною процесу розробки. Відладка є ключовою для виявлення та усунення помилок, що можуть виникнути в коді системи захисту. Вбудований відладчик PyCharm дозволяє розробникам відстежувати виконання програми, а також ефективно виявляти та виправляти помилки, забезпечуючи стабільну та надійну роботу системи..

2. Вбудований інтерпретатор Python, який забезпечує можливість виконання програм без необхідності їх компіляції, що робить процес розробки більш гнучким та швидким. Це особливо важливо в контексті розробки систем

захисту від DDoS-атак, де відразу важливо перевірити реакцію системи на реальних або симульованих даних.

3. Функція автодоповнення коду є додатковим інструментом, який полегшує процес написання коду. Розробникам необхідно менше часу для введення команд, оскільки IDE автоматично запропонує можливі варіанти завершення коду, що особливо корисно для розробки великих та складних систем захисту.

4. Можливість розширення та налаштування PyCharm робить його інструментом, який можна адаптувати під індивідуальні потреби розробників систем захисту. Це дає можливість створювати персоналізоване середовище розробки, враховуючи специфічні вимоги та особливості проекту.

В ході дослідження, нами було виявлено додаткові переваги використання PyCharm для розробки системи захисту від DDoS-атак [31]:

1. PyCharm є безкоштовним для особистого користування. Це робить його доступним для розробників з обмеженим бюджетом.

2. PyCharm має велику спільноту користувачів та розробників. Це означає, що розробники можуть отримати допомогу та підтримку від інших користувачів PyCharm.

Оглядаючи ці аспекти PyCharm, стає очевидним, що дане середовище розробки є дієвим інструментом для створення ефективних та надійних систем захисту від DDoS-атак. Завдяки своїм широким функціям та можливостям, PyCharm допомагає розробникам прискорити процес розробки, зробити його більш зручним та забезпечити високу якість програмного забезпечення.

PyCharm не лише надійне середовище розробки для Python, але й має широкий вибір плагінів, які розширюють його функціональність та роблять його ще потужнішим для розробки систем захисту від DDoS-атак. Тому ми розглянули кілька найбільш корисних плагінів [32], які можуть значно полегшити роботу розробників у цьому конкретному контексті:

1. PyCharm Security Plugin. Цей плагін спеціально розроблений для забезпечення безпеки програмного коду. Він включає інструменти аналізу коду

на предмет потенційних вразливостей, що може бути важливим для систем захисту від DDoS-атак. Плагін допомагає виявляти можливі слабкі місця в програмному коді та вчасно їх виправляти.

2. DDoS Mitigator. Цей плагін спрощує імітацію та тестування DDoS-атак на розроблюваній системі. Розробники можуть використовувати цей плагін для визначення, як їх система реагує на великі обсяги трафіку та атаки, а також для вдосконалення стратегій мітігації.

3. Performance Testing Plugin. Для ефективної системи захисту від DDoS-атак важливо визначити, наскільки добре вона працює під навантаженням. Цей плагін дозволяє розробникам виконувати тестування продуктивності для оцінки працездатності та виявлення можливих місць оптимізації.

4. Network Traffic Monitoring. Для систем захисту від DDoS-атак важливо слідкувати за мережевим трафіком. Цей плагін дозволяє розробникам в реальному часі спостерігати за мережевим трафіком, аналізувати його та виявляти аномалії, що може вказувати на можливі атаки.

5. Security Code Scan. Ще один плагін для забезпечення безпеки програмного коду. Він використовує алгоритми аналізу та перевірки на вразливості, допомагаючи виявляти та виправляти потенційні проблеми безпеки в коді системи захисту.

6. Load Testing Integration. Цей плагін інтегрується з популярними інструментами для тестування на витривалість та стресостійкість систем. Розробники можуть використовувати його для проведення тестів на велике навантаження та оцінки реакції системи на інтенсивний трафік.

7. Advanced Metrics Dashboard. Даний плагін надає розширений інтерфейс для моніторингу метрик продуктивності та стану системи. Зручна панель приладів дозволяє розробникам швидко отримувати інформацію про ключові параметри роботи системи захисту.

8. Automated Security Audits. Цей плагін автоматизує процес проведення аудитів безпеки коду. Він використовує стандарти та найкращі практики безпеки програмного забезпечення для визначення потенційних проблем та рекомендацій

щодо їх вирішення. Автоматизовані аудити можуть виявити слабкі місця в коді системи захисту та покращити її безпеку.

9. Vulnerability Assessment Toolkit. Цей плагін надає набір інструментів для оцінки вразливостей системи. Розробники можуть використовувати його для проведення сканування на предмет потенційних вразливостей, включаючи ті, які можуть бути використані в DDoS-атаках.

10. Interactive Threat Modeling. Плагін допомагає розробникам інтерактивно моделювати потенційні загрози та атаки на систему. Це може бути важливим для ефективного визначення стратегій захисту від DDoS та інших загроз безпеки.

11. Real-time Anomaly Detection. Даний плагін спрямований на реальний час виявлення аномалій у мережевому трафіку та системі в цілому. Використовуючи різноманітні алгоритми та правила, він дозволяє розпізнавати ненормальні патерни, що можуть свідчити про DDoS-атаки.

12. Integration with Security Frameworks. Плагін, який надає можливість інтеграції PyCharm з різними фреймворками та платформами для тестування безпеки. Це дозволяє розробникам використовувати різноманітні інструменти для перевірки безпеки своєї системи та реагувати на потенційні загрози.

13. Comprehensive Security Reports. Плагін, який автоматично генерує детальні звіти про стан безпеки системи. Це полегшує аналіз результатів тестів та аудитів безпеки, дозволяючи розробникам оперативно реагувати на виявлені проблеми.

14. Incident Response Toolkit. Плагін, спрямований на полегшення процесу реагування на інциденти. Він може автоматизувати деякі етапи реакції на DDoS-атаки, включаючи блокування атакуючих IP-адрес та відновлення роботи системи.

Загалом, ці плагіни розширюють базовий функціонал PyCharm, спрощуючи розробку та тестування систем захисту від DDoS-атак. Вони додають інструменти для виявлення та вирішення потенційних проблем безпеки, роблячи

PyCharm ще потужнішим інструментом для розробників, що працюють у сфері кібербезпеки.

2.3 Використання фреймворків Flask та Tkinter для розробки захисної програми

Для розробки захисної програми, яка включає в себе веб-інтерфейс для користувача, нами було обрано фреймворки – Flask (для реалізації веб-складової) та Tkinter (для створення графічного інтерфейсу користувача (GUI)), як ключові інструменти. Обираючи ці інструменти, ми врахували ряд ключових факторів, які роблять їх ідеальними для розробки захисної програми.

Flask для веб-інтерфейсу має наступні переваги [33]:

1. Легкість вивчення та спільнота. Flask визначається простим синтаксисом, який дозволяє швидко вивчити базові концепції. Його активна спільнота розробників забезпечує швидку підтримку та відповіді на запитання.
2. Масштабованість та гнучкість. Flask є легким та гнучким фреймворком, який може бути масштабованим від простих веб-додатків до складних систем. Це дозволяє створювати розширені та ефективні веб-інтерфейси для наших захисних програм.
3. Інтеграція з іншими інструментами. Flask легко інтегрується з іншими інструментами та бібліотеками, що дозволяє нам ефективно використовувати його для створення веб-сервісів та API (Application Programming Interface).

Tkinter для графічного інтерфейсу має наступні переваги [34]:

1. Простота та легкість використання. Tkinter є стандартним набором інструментів для створення GUI (Graphical User Interface) в Python. Він відзначається простим використанням та навчанням, що робить його ідеальним для розробників будь-якого рівня.

2. Крос-платформеність. Tkinter підтримується на різних операційних системах, включаючи Windows, macOS та Linux, забезпечуючи крос-платформену сумісність для нашого GUI (Graphical User Interface).

3. Можливості дизайну. Tkinter дозволяє створювати різноманітні та привабливі дизайни інтерфейсів, забезпечуючи гнучкість у вигляді та взаємодії з користувачем.

Загалом, вибір Flask та Tkinter обґрунтовується їхньою ефективністю, легкістю вивчення та можливістю масштабування. Ці інструменти допоможуть нам створити комплексну та ефективну захисну програму з високоякісним веб-інтерфейсом та GUI (Graphical User Interface).

Також, ми з’ясували кілька додаткових переваг використання Flask та Tkinter для розробки захисних програм [33, 34]:

1. Безкоштовність. Обидва інструменти є абсолютно безкоштовними для особистого та комерційного використання. Це значно спрощує доступність інструментів для розробників будь-якого рівня без необхідності плати за ліцензії або обмежень використання.

2. Документація та навчання. Як стандартні інструменти для розробки у світі Python, Flask та Tkinter мають обширну та детальну документацію. Це значно полегшує процес навчання і розуміння функціоналу обох інструментів. Розробники можуть ефективно використовувати ці ресурси для швидкого вивчення та вирішення проблем.

3. Розширюваність та налаштування. Flask та Tkinter є дуже розширюваними, що дозволяє розробникам додавати новий функціонал та налаштовувати їх під конкретні потреби проекту. Використання розширень Flask та можливостей Tkinter гарантує гнучкість у вигляді та функціоналу захисної програми.

4. Доступність інструментів. Розширення Flask та Tkinter є доступними для великої кількості розробників завдяки їх широкому застосуванню в галузі розробки. Це також означає наявність безлічі готових рішень, які спрощують вирішення типових задач та прискорюють розробку.

5. Велика активна спільнота. Спільнота користувачів та розробників Flask та Tkinter дуже активна. Це забезпечує швидку підтримку, вирішення проблем та обмін досвідом. Активні форуми, блоги та ресурси дозволяють розробникам бути в курсі останніх трендів та рекомендацій.

Загалом, ці переваги підсилюють вибір Flask та Tkinter як інструментів для розробки захисної програми, дозволяючи нам ефективно використовувати їхні можливості для створення комплексної та надійної захисної програми.

Ми визначили, що при використанні Flask та Tkinter для розробки захисних програм, слід враховувати кілька додаткових міркувань:

1. Безпека. Важливо вжити заходів безпеки для захисту захисної програми від злому. Це можна зробити, використовуючи такі методи, як шифрування, аутентифікація та авторизація. Flask та Tkinter пропонують деякі вбудовані функції безпеки, але, на нашу думку, розробникам слід також вжити додаткових заходів для захисту своєї системи. Наприклад, розробники можуть використовувати шифрування для захисту даних, що передаються між веб-сервером та GUI. Вони також можуть використовувати аутентифікацію та авторизацію для обмеження доступу до системи до авторизованих користувачів.

2. Надійність. Захисна програма повинна бути надійною, щоб вона могла ефективно виконувати свої функції. Flask та Tkinter є надійними інструментами, але, на нашу думку, розробникам слід вжити заходів для забезпечення надійності своєї системи. Наприклад, розробники можуть використовувати балансування навантаження для розподілу навантаження між декількома серверами. Вони також можуть використовувати резервне копіювання та відновлення для захисту своїх даних у разі збоїв.

3. Взаємодія Flask та Tkinter. Для ефективної взаємодії між веб-інтерфейсом (Flask) та графічним інтерфейсом користувача (Tkinter), важливо правильно налаштувати комунікацію між ними. Flask може використовувати RESTful API для обміну даними з іншими складовими програми. Забезпечення асинхронного обміну інформацією може покращити продуктивність та відзначити взаємодію з іншими компонентами. Інтерфейс Tkinter може

використовувати бібліотеки для HTTP-запитів або WebSocket для отримання даних від серверної частини. Важливо правильно обробляти асинхронні події та взаємодіяти з сервером без збоїв у графічному інтерфейсі.

4. Оновлення та підтримка. У контексті захисної програми важливо регулярно оновлювати і підтримувати як Flask, так і Tkinter. Періодичні оновлення Flask дозволяють використовувати нові можливості та виправляти потенційні уразливості. Забезпечення сумісності між версіями Flask важливо для збереження стабільності програми. Підтримка бібліотеки Tkinter забезпечується за допомогою оновлень та корекцій у нових версіях Python. Важливо слідкувати за оновленнями і використовувати найновіші версії для отримання найкращої підтримки та фіксації проблем.

Враховуючи ці міркування, розробники можуть ефективно використовувати Flask та Tkinter для створення безпечних, надійних та легко оновлюваних захисних програм з високоякісним інтерфейсом.

Отже, фреймворки Flask та Tkinter є потужними та гнучкими інструментами, які можна використовувати для розробки ефективної захисної програми. Завдяки своїм широким функціям та можливостям, Flask та Tkinter можуть допомогти розробникам швидко та ефективно створити, тестувати та запускати захисну програму.

2.4 Висновки до розділу 2

Другий розділ присвячено огляду ключових інструментів та технологій, які можуть бути використані для розробки системи захисту від DDoS-атак. На основі проведеного дослідження можна зробити наступні висновки:

1. Для розробки системи захисту від DDoS-атак ми вирішили використовувати мову програмування, яка має високу продуктивність і може ефективно обробляти великі обсяги даних. Python є популярним вибором для розробки систем захисту від DDoS-атак через ряд його переваг. Аналізуючи ряд

переваг, які надає Python, ми переконалися, що ця мова є ідеальним вибором для нашої програми, забезпечуючи ефективність, легкість розробки та високу масштабованість, не зважаючи на деякі обмеження у швидкодії та оптимізації пам'яті.

2. Для розробки системи захисту від DDoS-атак ми вирішили використовувати середовище розробки, яке надає широкий спектр функцій для полегшення та прискорення процесу розробки. PyCharm є хорошим вибором, оскільки має ряд функцій, які роблять його зручним для розробки систем захисту від DDoS-атак, а саме: вбудовані відладчик та інтерпретатор; функція автодоповнення коду; можливість розширення та налаштування. Це дозволяє розробникам швидко та ефективно створювати, тестувати та запускати системи захисту від DDoS-атак.

3. Для розробки веб-інтерфейсу для системи захисту від DDoS-атак ми вирішили використовувати фреймворк, який є простим у використанні і масштабованим. Flask є хорошим вибором для цієї мети. Для розробки графічного інтерфейсу користувача для системи захисту від DDoS-атак слід використовувати фреймворк, який є простим у використанні і забезпечує базові засоби безпеки. Tkinter є хорошим вибором для цієї мети. Такий вибір виник через кілька причин, а саме: їх популярність; активну спільноту користувачів та розробників; простоту вивчення; масштабованість і безкоштовність; їхню можливість розширення та налаштування, що дозволяють створювати надійні та безпечні системи захисту. В результаті це дозволяє розробникам швидко та ефективно розгортати захисні програми.

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМИ ЗАХИСТУ ВІД DDOS-АТАК

3.1 Проектування програми захисту від DDoS-атак

Проектування пакетної структури є першим і ключовим етапом у розробці програмного забезпечення. На цьому етапі була створена діаграма пакетів (рис. 3.1), що наочно відображає структуру системи захисту від DDoS-атак.

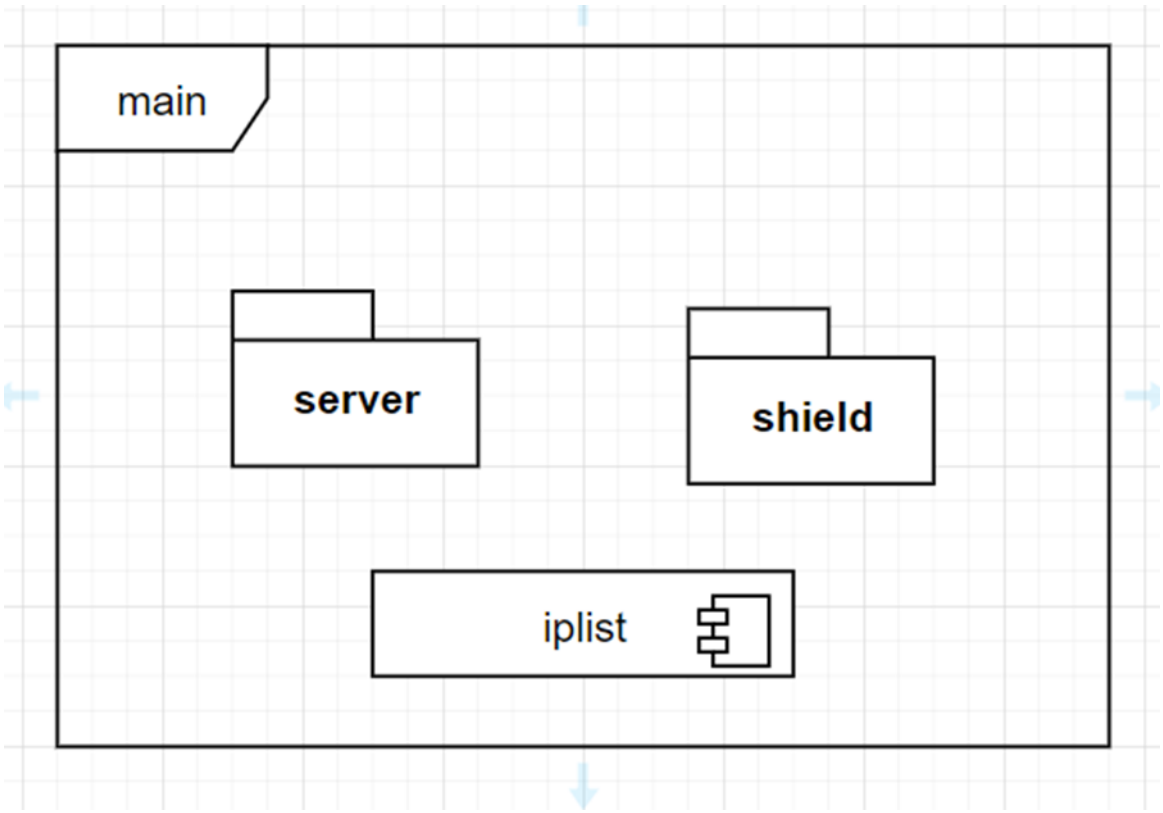

Рис. 3.1. Діаграма пакетів

Діаграма пакетів [35] використовує уніфіковану мову моделювання для відображення взаємозв'язків між ключовими компонентами системи. Окрім стандартних відношень залежності UML (Unified Modeling Language), використовуються два спеціальні типи залежностей: імпорт пакетів та злиття пакетів.

Імпорт пакетів вказує на зв'язок між простором імен імпорту та пакетом, дозволяючи додавати імена членів пакета до власного простору імен.

Злиття пакетів вказує на об'єднання вмісту двох пакетів. Це застосовується, коли елемент існує як у вихідному, так і в цільовому пакеті.

Діаграми пакетів використовуються для ілюстрації функціональності та багатошарової архітектури програмної системи. Вони дозволяють визначити структуру та взаємозв'язки між компонентами системи.

Під час проектування програмного забезпечення зазвичай розглядають і внутрішню структуру системи, яка відображає взаємодію між класами і модулями проекту, надаючи зрозумілий огляд їхньої структури.

1. Flask-додаток для обробки запитів. Він відповідає за прийом та обробку HTTP-запитів. Основні компоненти цього модуля:

– Маршрутизація запитів. Flask визначає кілька маршрутів, таких як головна сторінка ("/") та сторінка з виведенням списку IP-адрес. Це визначається за допомогою декораторів, що забезпечують відповідний обробник для кожного маршруту.

– Обробка запитів. При отриманні запиту Flask передає його відповідному обробнику, який виконує необхідні дії, такі як отримання IP-адреси з заголовків запиту та передача її до об'єкта `IpHandler`.

2. Клас IpHandler. Він відповідає за обробку IP-адрес та управління списком. Основні функції та взаємодія:

– Додавання IP-адрес до списку. Клас отримує IP-адрес від Flask-додатку та визначає, чи він повинен бути доданий до списку. Якщо IP-адрес вже присутній, клас вирішує, чи слід його додати до файлу через метод `write_to_file`.

– Запис до файлу. Якщо кількість входжень певного IP-адресу перевищує задане значення, клас викликає метод `write_to_file`, який записує цей IP-адрес до файлу.

3. Клас FileHandler. Він відповідає за роботу з файловою системою та управління списком IP-адрес. Основні функції та взаємодія:

– Відкриття файлу. Клас може відкривати файл із збереженими IP-адресами для подальшого використання.

– Збереження файлу. Клас може зберігати змінений список IP-адрес у визначений файл для подальшого використання.

4. Tkinter GUI (TextEditorApp). Графічний інтерфейс Tkinter дозволяє користувачеві взаємодіяти зі списком IP-адрес та файловою системою. Основні функції та взаємодія:

– Кнопки для відкриття та збереження файлу. Користувач може взаємодіяти з файловою системою, відкриваючи та зберігаючи список IP-адрес.

– Додавання та видалення IP-адрес. Кнопки для додавання нових IP-адрес та видалення існуючих.

– Редагування списку. Текстовий редактор дозволяє користувачеві зручно редагувати список IP-адрес.

5. Головний клас TextEditorApp. Цей клас є центральним елементом графічного інтерфейсу і відповідає за відображення та взаємодію користувача зі списком IP-адрес. Основні функції та взаємодія:

– Ініціалізація інтерфейсу. Клас ініціалізує головне вікно, створює розділ для кнопок та текстового редактора.

– Відкриття файлу. Забезпечує функціонал для відкриття файлу з вмістом IP-адрес та відображення їх у текстовому редакторі.

– Збереження файлу. Забезпечує функціонал для збереження змін у текстовому редакторі у визначений файл.

– Додавання та видалення IP-адрес. Надає можливість користувачеві взаємодіяти зі списком, додаючи нові або видаляючи існуючі IP-адреси.

– Редагування списку. Забезпечує можливість користувачеві зручно редагувати список IP-адрес.

– Виклик функцій IpHandler та FileHandler. Під час додавання, видалення та редагування IP-адрес, відбувається взаємодія з об'єктами `IpHandler` та `FileHandler`.

6. Головне вікно та елементи інтерфейсу:

– Статусна мітка. Інформує користувача про стан захисту системи - увімкнено чи вимкнено.

– Кнопка "Toggle Protection". Дозволяє увімкнути та вимкнути захист системи від DDoS-атак.

– Поток Flask. Взаємодія з Flask відбувається в окремому потоці (`flask_thread`). Це дозволяє обслуговувати HTTP-запити паралельно з роботою графічного інтерфейсу.

7. Оптимізація та заходи безпеки:

– Захист від DDoS-атак. Flask Limiter використовується для обмеження кількості HTTP-запитів. Його стан можна змінити за допомогою кнопки "Toggle Protection".

– Обмеження кількості IP-адрес. Система обмежує кількість IP-адрес у списку та в файлі для збереження ефективності та безпеки.

– Відслідковування IP-адрес. Клас `IpHandler` відстежує кількість входжень кожного IP-адресу та, при необхідності, записує його в файл для подальшого аналізу.

Ці компоненти та їхні взаємодії розглядаються як цілісна система, яка забезпечує ефективне виявлення та захист від DDoS-атак, а також зручне управління та редагування списком IP-адрес.

3.2 Архітектура розробки програми захисту від DDoS-атак

Архітектура розробки програми [36] захисту від DDoS-атак є важливим етапом при проектуванні програми, оскільки допомагає організувати код та функціональність програми, щоб забезпечити ефективність та зрозумілість розробникам. Для візуалізації та розуміння основних етапів та функцій програми захисту від DDoS-атак, було створено блок-схему програми (рис. 3.2), яка відображає послідовність операцій та контрольних пунктів, які програма виконує для виявлення, блокування та мінімізації впливу DDoS-атак на веб-ресурс.

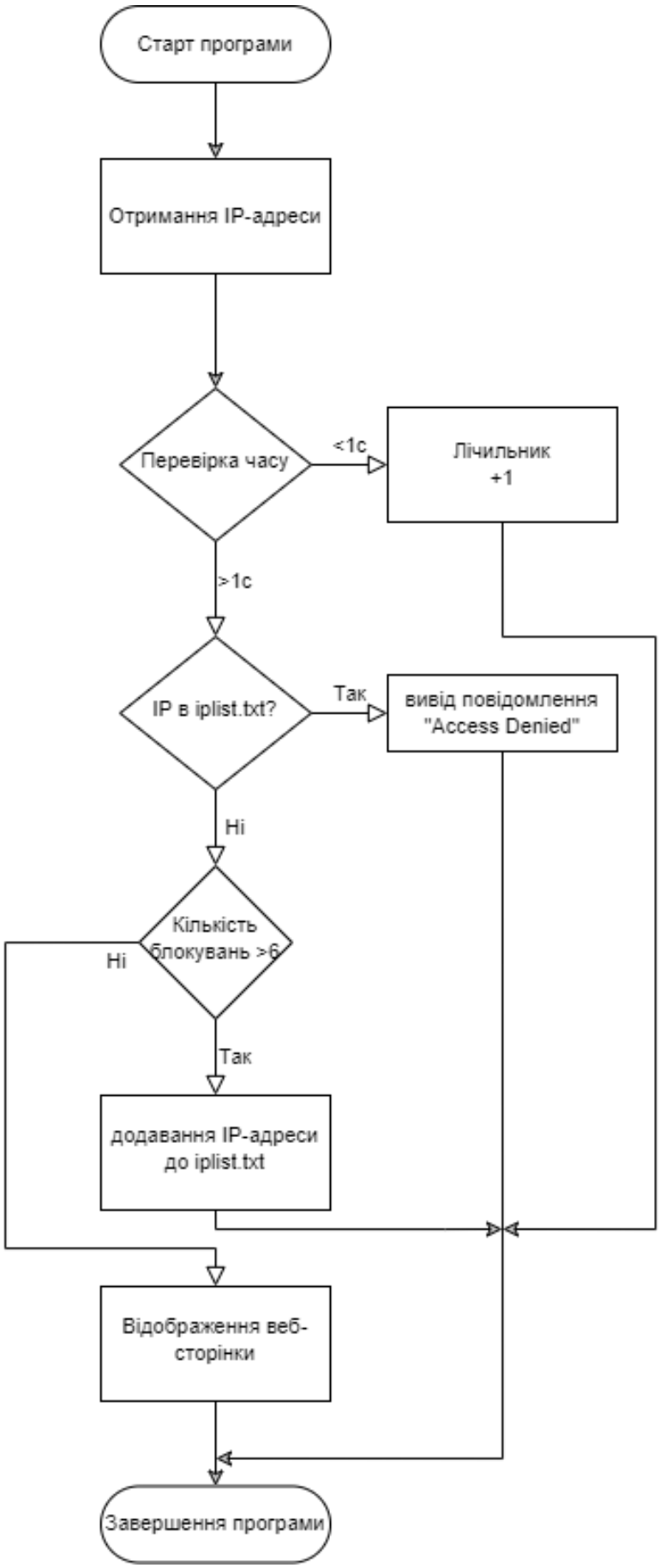

Рис. 3.2. Блок-схема програми захисту від DDoS-атак

Дана блок-схема показує загальний огляд програми захисту від DDoS-атак. За блок-схемою, що наведена на рис. 3.2, розробники в свої подальшій роботі мають реалізувати такі функції:

1. Основний сервер відповідає за обробку запитів від користувачів і надання веб-сторінки текстового редактора.

2. Щит захищає основний сервер від DDoS-атак, обмежуючи кількість запитів за секунду.

3. Обробник IP-адрес підтримує список IP-адрес, які отримували доступ до сервера. Він додає нові IP-адреси до списку та блокує IP-адреси, які зробили занадто багато запитів.

4. Текстовий редактор дозволяє користувачам відкривати, зберігати та редагувати список IP-адрес.

Крім того, блок-схема та майбутній код мають узгоджуватися в таких аспектах:

1. Основний сервер представлений класом Flask у коді.

2. Щит представлений класом Limiter у коді.

3. Обробник IP-адрес представлений класом IpHandler у коді.

4. Текстовий редактор представлений класом TextEditorApp у коді.

Блок-схема також показує такі взаємодії між різними компонентами:

1. Основний сервер спілкується з обробником IP-адрес, щоб перевірити, чи заблокована IP-адреса клієнта.

2. Обробник IP-адрес спілкується з обробником файлів для читання та запису списку IP-адрес.

3. Текстовий редактор спілкується з обробником IP-адрес для читання та запису списку IP-адрес.

Тому необхідно, щоб розробники реалізує ці взаємодії у майбутньому коді таким чином:

1. Основний сервер використовує метод IpHandler.add_ip_address() для перевірки, чи заблокована IP-адреса клієнта.

2. Обробник IP-адрес використовує методи FileHandler.open_file() і FileHandler.save_file() для читання та запису списку IP-адрес.

3. Текстовий редактор використовує методи IpHandler.add_ip_address(), IpHandler.delete_ip_address() і IpHandler.get_ip_addresses() для читання та запису списку IP-адрес.

Загалом, блок-схема, що представлена на рис. 3.2 та майбутній код мають відповідати один одному та точно представляти програму захисту від DDoS-атак.

3.3 Розробка програми захисту від DDoS-атак

Розроблена програма захисту від DDoS-атак включає в себе веб-застосунок (Flask) та графічний інтерфейс користувача (Tkinter). Основна мета цієї програми – реалізація системи захисту від DDoS-атак та одночасно надання можливості редагування та збереження списку IP-адрес.

Розглянемо детальніше класи та функції, що були використані у розробленій програмі:

1. Клас `IpHandler` має наступні функції (рис. 3.3):

– `write_to_file(ip)` - статичний метод, який додає переданий IP-адрес до файлу "iplist.txt". Якщо виникає помилка при записі в файл, виводиться повідомлення про помилку.

– `add_ip_address(ip, ip_addresses)` - статичний метод, який додає переданий IP-адрес до списку `ip_addresses`. Якщо IP-адрес ще не існує в списку, він додається. Якщо IP-адрес зустрічається більше 30 разів, він додається до файлу через метод `write_to_file`. Якщо список перевищує розмір 10 000 елементів, видаляється перший елемент.

```python
4 usages
class IpHandler:
    FILE_PATH = "B://Dyploma/iplist.txt"

    1 usage
    @staticmethod
    def write_to_file(ip):
        try:
            with open(IpHandler.FILE_PATH, 'a') as file:
                file.write(ip + "\n")
            print(f"IP {ip} added to the file.")
        except Exception as e:
            print(f"Error writing to file: {e}")

    1 usage
    @staticmethod
    def add_ip_address(ip, ip_addresses):
        print(f"Processing IP: {ip}")

        if ip not in ip_addresses:
            ip_addresses.append(ip)
            print(f"IP {ip} added to the list.")
        elif ip_addresses.count(ip) > 30:
            print(f"IP {ip} added to the file.")
            IpHandler.write_to_file(ip)

        if len(ip_addresses) > 10000:
            ip_addresses.pop(0)
```

Рис. 3.3. Клас `IpHandler`

2. Клас `FileHandler` має наступні функції (рис. 3.4):

– `open_file()` - статичний метод, який відкриває файл "iplist.txt" та повертає його вміст. Якщо файл не знайдений, повертається пустий список.

– `save_file(content)` - статичний метод, який зберігає переданий вміст у файл "iplist.txt". Якщо виникає помилка при збереженні файлу, виводиться повідомлення про помилку.

```
3 usages
class FileHandler:
    FILE_PATH = "B://Dyploma/iplist.txt"

    1 usage
    @staticmethod
    def open_file():
        try:
            with open(FileHandler.FILE_PATH, 'r') as file:
                content = file.readlines()
            return content
        except FileNotFoundError:
            return []

    1 usage
    @staticmethod
    def save_file(content):
        try:
            with open(FileHandler.FILE_PATH, 'w') as file:
                file.writelines(content)
        except Exception as e:
            print(f"Error saving file: {e}")
```

Рис. 3.4. Клас `FileHandler`

3. Flask-додаток має наступну функцію:

- `@app.route('/')` (рис. 3.5) - декоратор, який встановлює маршрут для головної сторінки. Обробник маршруту викликає `index()` - функцію, яка отримує IP-адрес користувача, додає його до списку `ip_addresses` та відображає сторінку "index.html" зі списком IP-адрес.

```
@app.route('/')
@limiter.limit("5 per second")
def index():
    ip_address = request.headers.get( key: 'X-Forwarded-For', request.remote_addr)
    IpHandler.add_ip_address(ip_address, ip_addresses)
    return render_template( template_name_or_list: 'index.html', ip_addresses=ip_addresses)
```

Рис. 3.5. Функція `@app.route('/')`

Основний Flask-додаток має лише один маршрут /, який відповідає функції index. Цей маршрут обробляє запити до кореневого шляху веб-застосунку. Він використовує Flask Limiter для обмеження частоти запитів до 5 за секунду.

4. Клас `TextEditorApp` має наступні функції (рис. 3.6):

– `open_file()`- метод, який викликається при натисканні кнопки "Open IpList". Відкриває вміст файлу "iplist.txt" та відображає його в текстовому віджеті.

– `save_file()` - метод, який викликається при натисканні кнопки "Save IpList". Зберігає вміст текстового віджету у файл "iplist.txt".

– `add_text()` - метод, який викликається при натисканні кнопки "Add ip". Додає новий текст з поля вводу до текстового віджету.

– `delete_text()` - метод, який викликається при натисканні кнопки "Delete ip". Видаляє введений текст з текстового віджету.

– `remove_empty_lines()` - метод, який видаляє порожні рядки з текстового віджету.

– `display_content(content)` - метод, який відображає переданий вміст в текстовому віджеті.

```
1 usage
class TextEditorApp:
    def __init__(self, root):...
    1 usage
    def open_file(self):...
    1 usage
    def save_file(self):...
    1 usage
    def add_text(self):...
    1 usage
    def delete_text(self):...
    1 usage
    def remove_empty_lines(self):...
    1 usage
    def display_content(self, content):...
```

Рис. 3.6. Клас `TextEditorApp`

5. Інші функції (рис. 3.7):

– `run_flask_in_thread()` - функція, яка запускає Flask-додаток у окремому потоці.

– `toggle_protection()` - функція, яка перемикає захист від DDoS-атак.

– `on_closing()` - функція, яка викликається при закритті вікна та вимикає захист.

```
1 usage
def run_flask_in_thread():
    app.run(debug=False, host='0.0.0.0')

1 usage
def toggle_protection():
    if limiter.enabled:
        limiter.enabled = False
        status_label.config(text="Protection Disabled")
    else:
        limiter.enabled = True
        status_label.config(text="Protection Enabled")

1 usage
def on_closing():
    limiter.enabled = False
    root.destroy()
```

Рис. 3.7. Інші функції

6. Код також містить графічний інтерфейс Tkinter для взаємодії з користувачем та відображення стану захисту від DDoS-атак.

Розроблена програма реалізує захист від DDoS-атак (Flask Limiter) наступним чином:

– Використовує бібліотеку Flask Limiter для обмеження кількості запитів до веб-застосунку.

– Встановлено ліміт "5 запитів за секунду" для маршруту `/`.

– Включає можливість вручну вмикати/вимикати захист від DDoS-атак через графічний інтерфейс Tkinter.

Загалом, розроблена програма для досягнення своєї мети виконує наступну послідовність дій:

1. Користувач відкриває програму та бачить графічний інтерфейс Tkinter.
2. Програма визначає IP-адресу користувача та додає її до списку IP-адрес.
3. Користувач може вручну редагувати, додавати або видаляти IP-адреси через графічний інтерфейс.
4. Список IP-адрес може бути збережений у файл або відкритий з файлу.
5. Програма забезпечує захист від DDoS-атак, обмежуючи кількість запитів до сервера.

Отже, написаний нами код створює комбінований інструмент для редагування списку IP-адрес та забезпечує простий захист від DDoS-атак на рівні веб-застосунку.

3.4 Планування процесу розгортання системи

Для ілюстрації фізичного розташування компонентів системи використовується діаграма розгортання, побудована з використанням Unified Modeling Language (UML).

Діаграма розгортання [37] є потужним інструментом для візуалізації фізичного розгортання артефактів на вузлах системи. У контексті нашого дослідження ця діаграма використовується для детального опису розташування апаратних і програмних компонентів системи захисту від DDoS-атак.

Схема розгортання системи наведена на рис. 3.8.

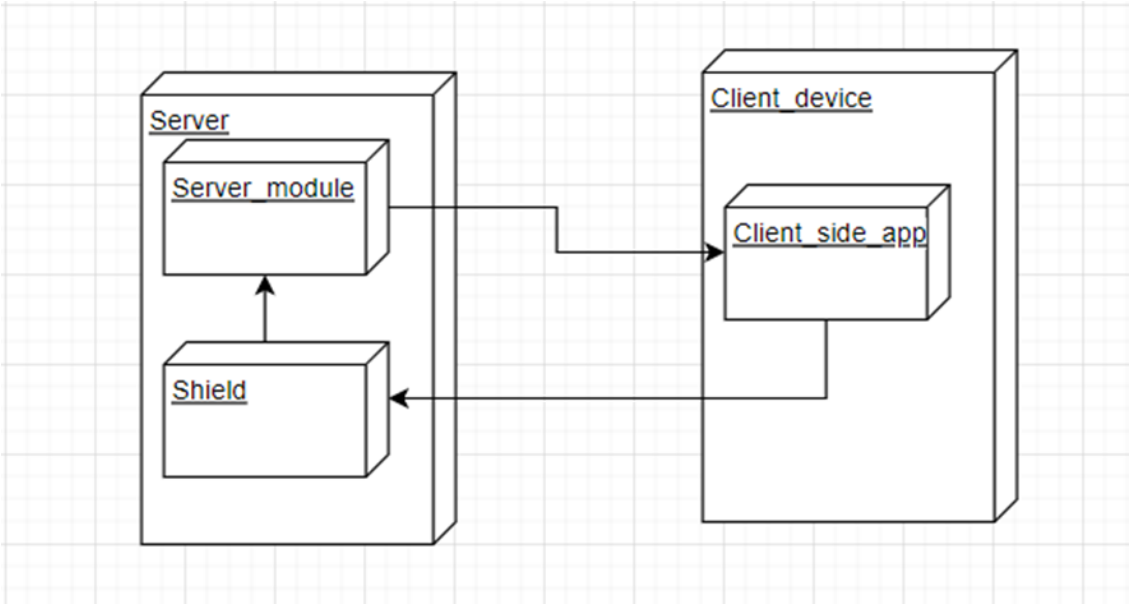

Рис. 3.8. Діаграма розгортання системи

На даній діаграмі відображені:

1. Вузли пристроїв. Вони представляють фізичні обчислювальні ресурси, такі як сервери, які використовуються для виконання програмного забезпечення системи захисту. Кожен вузол пристрою може включати в себе підвузли, визначаючи, наприклад, кластер серверів баз даних.

2. Вузли середовищ виконання (EEN). Ці вузли представляють середовища, де виконуються програми системи. Вони можуть включати в себе віртуальні машини та інші абстракції, які забезпечують виконання програмного коду.

Загалом, діаграма розгортання надає повний огляд архітектури системи та показує взаємодію між фізичними та програмними компонентами.

3.5 Тестування розробленої програми захисту від DDoS-атак

Програмне тестування [38] – це процес перевірки артефактів та поведінки програмного забезпечення для визначення його правильності та ефективності. Воно надає об'єктивну, незалежну оцінку програми, що дозволяє бізнесу оцінити ризики, пов'язані з її впровадженням.

Методи тестування включають аналіз вимог до продукту, перегляд архітектури та дизайну, співпрацю з розробниками для удосконалення кодування, виконання програми для перевірки її поведінки, перевірку інфраструктури розгортання, участь у виробничій діяльності та інші методи.

Тестування програмного забезпечення дозволяє отримати об'єктивну інформацію про якість програми та ризик її збою.

Помилки виникають через процес кодування, що може призвести до дефектів у вихідному коді. Якщо ці дефекти виконані, система може неправильно працювати, що призведе до збою.

Не всі дефекти викликають збої; деякі можуть залишитися непоміченими, але при зміні середовища можуть призвести до збою. Переважною причиною дефектів є прогалини в вимогах, які можуть бути нефункціональними, такими як тестованість, масштабованість та інші.

Тестування може бути статичним [39], де використовуються огляди та аналіз вихідного коду, динамічним [39], де тестується виконання програми, та пасивним, коли перевіряються системні журнали та трасування без взаємодії з програмою.

Дослідницьке тестування [40] – це підхід, що поєднує навчання, розробку та виконання тестів. Воно підкреслює особисту відповідальність тестувальника за постійне вдосконалення якості роботи.

У даній роботі ми використали метод тестування з функціональною точкою зору, який оцінює правильність та ефективність функцій нашої програми. Ми здійснили тестування обмеження кількості запитів, обмеження та додавання IP-адрес до iplist, а також обмеження доступу до сервера для IP-адрес, які містяться в iplist.

Такий підхід дозволив перевірити, чи працюють встановлені обмеження, як вони реагують на DDoS-атаки, і чи правильно виконується додавання IP-адрес до iplist. Також перевірили доступ до сервера для IP-адрес, які знаходяться в iplist, що дозволяє нам переконатися, що цей механізм обмеження доступу працює належним чином.

Загалом, ми протестували функціональність програми в умовах реального використання та атак, щоб переконатися, що вона працює згідно з очікуваннями та має відповідний рівень захисту.

Під час тестування програми, ми провели декілька тестів:

1. Тест № 1, який полягає в обмеженні кількості запитів.
2. Тест № 2, який полягає в обмеженні та додаванні IP-адрес до iplist.
3. Тест № 3, який полягає в обмеженні доступу до сервера, якщо IP-адреса є в списку iplist.

Розглянемо детальніше, як проводилося кожне тестування.

Тест № 1 мав кілька етапів:

1. Ми запустили локальний-сервер:

– перше відображення сторінки, що зображено на рис. 3.9, має текст "The student's local server Bezghubenko" і список IP-адрес, який змінюється при кожному відвідуванні сторінки, що відображено на рис. 3.10.

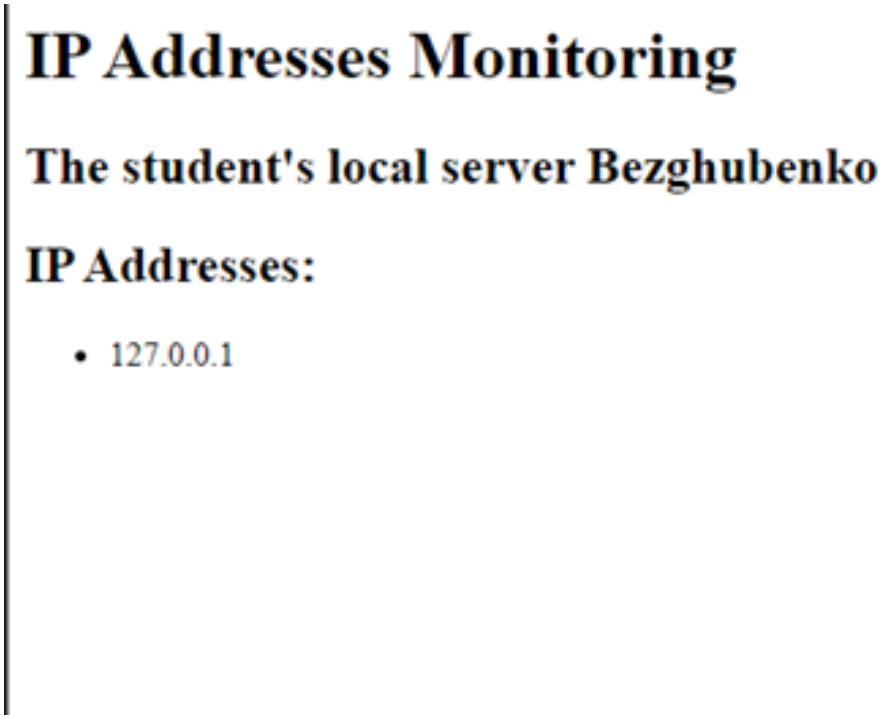

Рис. 3.9. Головна сторінка локального серверу

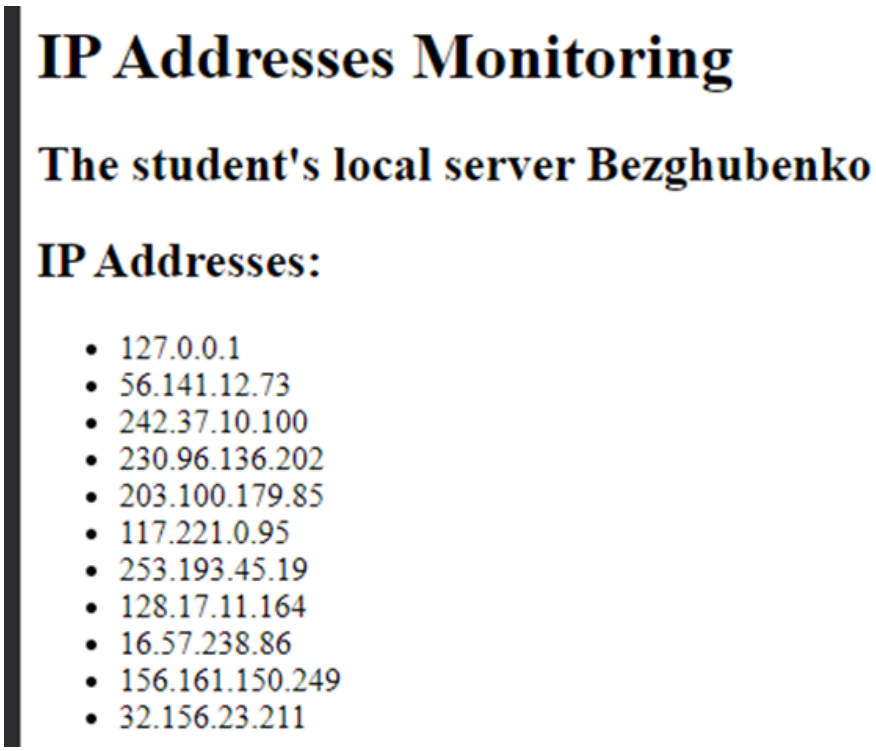

Рис. 3.10. IP-адреси, які відвідали сторінку

2. Ми активували захист:

– активація захисту здійснюється натисканням кнопки "Toggle Protection", яка використовує `flask_limiter` для обмеження кількості запитів в секунду, а також відкритий IpList, який є пустим (рис. 3.11).

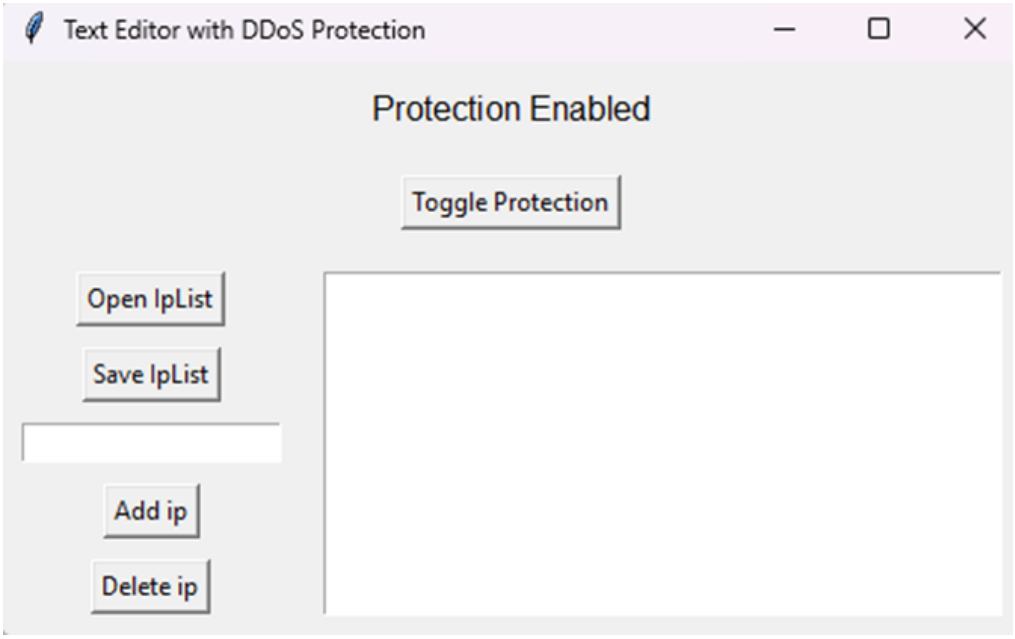

Рис. 3.11. Демонстрація IpList, який є пустим під час активації захисту

3. Ми провели DDoS-атаку:

– провели DDoS-атаку, яка надсилала 10 запитів від кількох IP-адрес. Результатом є отримання повідомлення "429 Too Many Requests", що свідчить про те, що система успішно обробила велику кількість запитів та відхилила їх (рис. 3.12). Також IP-адреси не було внесено в IpList (рис. 3.13).

Рис. 3.12. Отримане повідомлення

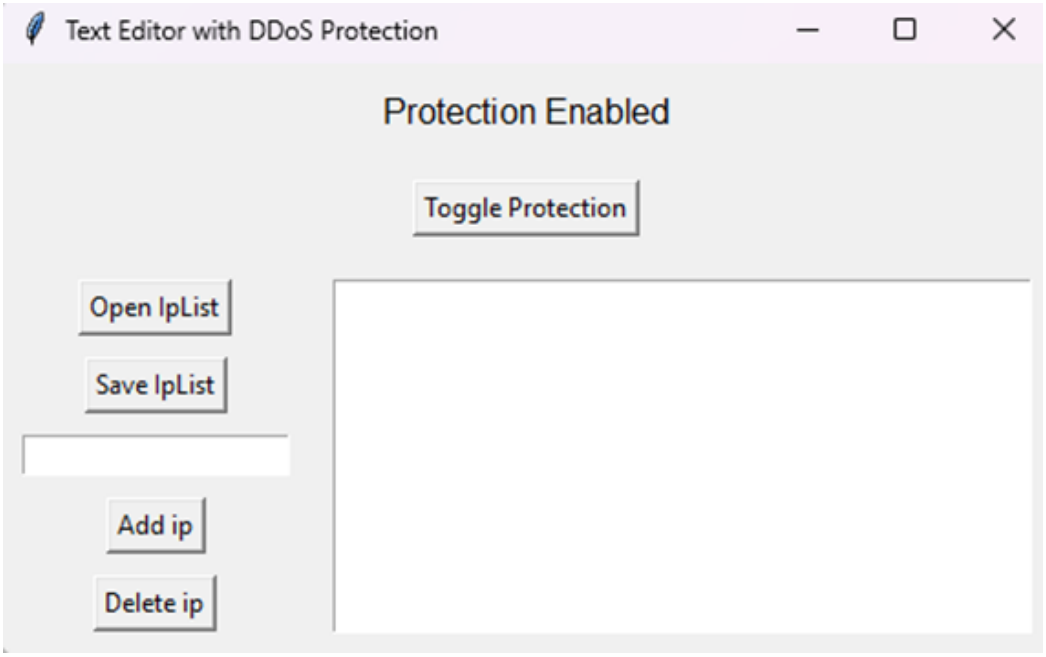

Рис. 3.13. Демонстрація IpList, який є пустим після DDoS-атаки

Під час тесту № 2 ми провели більш складнішу атаку з більшою кількістю IP-адрес, які надсилають по 300 запитів. В результаті IP-адреси блокуються 6 разів і через це заносяться в IpList. На рис. 3.14 відображено, які IP-адреси підключалися та надсилали запити, та ці ж адреси додані до IpList.

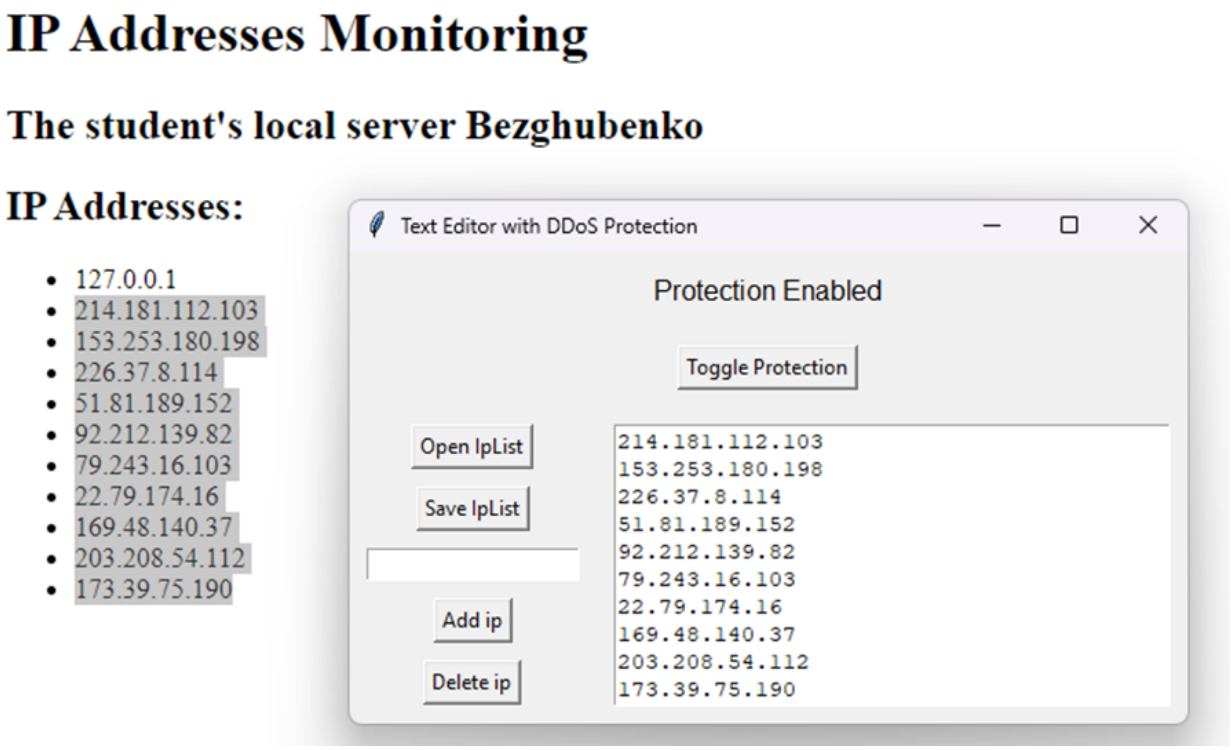

Рис. 3.14. Демонстрація заблокованих IP-адрес в IpList

Під час тесту № 3 ми додали IP-адресу в IpList та перевірили чи має вона доступ. В результаті було відображено відмову в доступі до серверу (рис. 3.15).

Рис. 3.15. Демонстрація відмови в доступі до серверу

3.6 Висновки до розділу 3

Третій розділ присвячено проектуванню та розробці програми захисту від DDoS-атак. В результаті проведеного дослідження:

1. Було ретельно розглянуто проектування програми захисту від DDoS-атак, а саме пакетну структуру та внутрішню структуру системи. В підсумку нам вдалося відобразити ключові компоненти майбутньої програми, їхню взаємодію та функціональність.

2. Було розглянуто архітектуру розробки програми захисту від DDoS-атак. Для візуалізації основних етапів та функцій програми було створено блок-схему, яка відображає послідовність операцій та контрольні пункти для виявлення, блокування та мінімізації впливу DDoS-атак. В підсумку це сприяє організації коду та функціоналу для досягнення ефективності та зрозумілості для розробників майбутньої програми.

3. Було описано розроблену програму, основна мета якої полягає у реалізації системи захисту від DDoS-атак та одночасно наданні можливості редагування та збереження списку IP-адрес. В результаті ми створили програму, яка є комбінованим інструментом для редагування списку IP-адрес та забезпечує простий захист від DDoS-атак на рівні веб-застосунку.

4. Було здійснено планування процесу розгортання системи, в результаті чого була створена діаграма розгортання за допомогою UML. Ця діаграма призначена для докладного аналізу фізичного розташування компонентів системи захисту, зокрема апаратних і програмних елементів.

5. Було проведено тестування розробленої програми захисту від DDoS-атак, використовуючи метод функціональної точки зору, для оцінки її функціональності та ефективності. Тестування проводилися в кілька етапів, а саме: обмеження кількості запитів, додавання IP-адрес до списку блокування та обмеження доступу для IP-адрес із цього списку. Отримані результати підтверджують успішну роботу захисної системи, виявляючи її здатність відхиляти DDoS-атаки та ефективно управляти обмеженням доступу до сервера.

РОЗДІЛ 4.
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Екологічне законодавство України представляє собою комплексну систему правових актів, що включають в себе норми еколого-правового характеру. Ці норми спроможні регулювати відносини, пов'язані з використанням природних ресурсів, охороною природного середовища та забезпеченням екологічної безпеки.

Екологічне законодавство України оформлено в такій структурі:

- Загальні положення і принципи;
- Земельне законодавство;
- Водне законодавство;
- Лісове законодавство;
- Гірське законодавство;
- Фауністичне законодавство;
- Законодавство про охорону й використання атмосферного повітря;
- Законодавство про природні території і об'єкти особливої охорони;
- Законодавство про екологічну безпеку.

Основні положення екологічного законодавства визначені у Конституції України [1], прийнятій у 1996 році.

З 159 статей Конституції [1], 15 статей прямо чи опосередковано стосуються охорони природи, екологічної безпеки і природокористування. Ці статті (3, 13, 14, 16, 49, 50, 66, 85, 92, 106, 116, 119, 132, 138, 142) [1] містять основні екологічні права та обов'язки громадян, а також визначають повноваження законодавчої і виконавчої влади. Таким чином, в Україні вперше створено стійку основу для побудови системи екологічного законодавства.

Ієрархію нормативно-правових актів можна подати наступним чином:

1. Міжнародні конвенції та угоди, прийняті Верховною Радою України;
2. Закони України;

3. Підзаконні акти Верховної Ради України;

4. Укази Президента України;

5. Нормативно-правові акти Кабінету Міністрів України;

6. Загальнообов'язкові акти Міністерства екології та природних ресурсів, інших міністерств і відомств;

7. Відомчі акти органів державного управління.

Екологічні правовідносини регулюються також еколого-правовими нормами інших галузей законодавства, таких як цивільне, адміністративне, кримінальне тощо. Ці норми визначають основи та особливості притягнення осіб, винних у екологічних правопорушеннях, до відповідальності за дисциплінарним, адміністративним, майновим та кримінальним шляхом. Вони враховують ступінь провини, екологічний ризик, а також ступінь суспільної й екологічної небезпеки дій фізичних і юридичних осіб.

Кількість чинних нормативно-правових актів в країні постійно зростає. На сьогоднішній день в силі знаходиться понад 400 актів екологічного законодавства, включаючи близько 80 законів і постанов Верховної Ради України, приблизно 40 Указів Президента та більше 250 постанов Кабінету Міністрів України. Для забезпечення їх виконання було видано кілька сотень наказів міністерств і відомств.

Першим і найбільш узагальненим, а також ключовим за значенням законом, на якому базується вся система екологічного законодавства України, є Закон України «Про охорону навколишнього природного середовища» № 1264-XII [41], прийнятий 25 червня 1991 року.

У преамбулі та статті 1 Закону № 1264-XII [41] визначені цілі та завдання екологічного законодавства. Ці завдання спрямовані на реалізацію екологічної політики, спрямованої на збереження безпечної для існування живої і неживої природи навколишнього середовища, захист життя і здоров'я населення від негативного впливу, що виникає внаслідок забруднення природи. Також вони визначають мету досягнення гармонійної взаємодії між суспільством і природою, а також охорону, раціональне використання та відтворення природних ресурсів.

Екологічне законодавство, спрямоване на досягнення цих цілей, регулює суспільні відносини, пов'язані з об'єктами правової охорони, що визначені у статті 5 Закону № 1264-XII [41]. Ці об'єкти включають всі компоненти навколишнього природного середовища та усі природні ресурси. Повноваження законодавчої і виконавчої влади в сфері охорони навколишнього природного середовища закріплені у відповідних розділах III і IV Закону № 1264-XII [41].

У Законі № 1264-XII [41] також визначено положення щодо: спостереження, прогнозування, обліку та інформування; екологічної експертизи; стандартизації та нормування; контролю і нагляду; регулювання використання природних ресурсів; економічного механізму охорони навколишнього природного середовища; заходів екологічної безпеки; охоронюваних територій; надзвичайних екологічних ситуацій; спорів і відповідальності;- міжнародних відносин.

Загальні положення Закону № 1264-XII отримали розвиток у Законі України «Про оцінку впливу на довкілля» (2017 р.) [42], в окремих розділах інших законів: «Основи законодавства України про охорону здоров'я» (1992 р.) [43], «Про підприємництво» (1991 р.) [44], «Про інформацію» (1992 р.) [45], у Податковому кодексі України [46]; у Кодексі України про адміністративні правопорушення [47], в Кримінальному кодексі України [48] та інших законодавчих актах.

З тією самою метою, Кабінет Міністрів ухвалив ряд підзаконних актів. Серед них визначено положення про порядок видачі дозволів на спеціальне використання природних ресурсів, встановлення лімітів їх використання, положення про державний моніторинг навколишнього природного середовища. Декілька постанов Кабінету Міністрів регламентує діяльність екологічної інспекції.

Головним нормативним актом, що регулює відносини у сфері землекористування, є Земельний кодекс України [49]. Його основна мета полягає в створенні умов для раціонального використання та охорони земель, рівноправного розвитку усіх форм власності на землю і господарювання,

збереження та відновлення родючості ґрунтів, покращення природного середовища та захисту прав громадян, підприємств, установ і організацій щодо використання земель.

В Україні водні відносини регулюються Водним кодексом [50], прийнятим у 1995 році, а також іншими законодавчими актами. Головна мета цих нормативних актів полягає в забезпеченні збереження вод, науково обґрунтованого та раціонального використання водних ресурсів, відновлення водних ресурсів, захист вод від забруднення, засмічення та виснаження, уникнення негативного впливу на водні об'єкти та ліквідацію можливих наслідків шкідливих дій. Основна мета також включає покращення стану водних об'єктів, а також захист прав користувачів водних ресурсів.

Лісові відносини в Україні визначаються положеннями Лісового Кодексу [51], прийнятого у 1994 році, та іншими юридичними актами. Основна мета цих нормативів полягає в досягненні підвищення продуктивності праці, забезпеченні охорони та відновлення лісів, посиленні їх корисних властивостей і задоволенні потреб суспільства в лісових ресурсах шляхом їх науково обґрунтованого та раціонального використання.

В Україні гірничі відносини регулюються Кодексом України про надра [52], прийнятим у 1994 році, Гірничим законом України [53], ухваленим у 1999 році, та іншими законодавчими актами. Головна мета цих нормативних актів полягає в забезпеченні раціонального та комплексного використання надр для задоволення потреб у мінеральній сировині та інших суспільних виробничих потреб, а також в охороні надр, забезпеченні безпеки людей, майна та природи під час експлуатації надр. Крім того, ці нормативи спрямовані на захист прав і законних інтересів підприємств, установ, організацій і громадян у сфері використання надр.

Відносини у сфері охорони, використання та відтворення тваринного світу на території України (за винятком домашніх тварин) регулюються Законом України «Про тваринний світ» [54], прийнятим у 2001 році, а також іншими законодавчими актами. Головна мета цих нормативів полягає в збереженні та

поліпшенні умов існування диких тварин, забезпеченні сталого існування всіх видових та популяційних характеристик тварин у природному середовищі, в неволі або напіввільних умовах.

Закон України «Про охорону атмосферного повітря» (1992 р.) [55] визначає правові, організаційні основи та екологічні вимоги у сфері охорони атмосферного повітря. Основна мета цього закону полягає в збереженні та відновленні природного стану атмосферного повітря, створенні сприятливих умов для життєдіяльності, забезпеченні екологічної безпеки та запобіганні шкідливому впливу атмосферного повітря на здоров'я людей та навколишнє природне середовище.

Природоохоронне законодавство України, яке охоплює правові аспекти природних територій і об'єктів особливої охорони, включає в себе такі Закони України, як «Про природно-заповідний фонд України» (1992 р.) [56], «Про виключну (морську) економічну зону України» (1995 р.) [57], та «Про Червону книгу України» (2002 р.) [58], та інші нормативні акти. Ці документи визначають правові принципи природоохоронної діяльності на територіях (акваторіях), які є об'єктами особливої охорони. В них врегульовані питання класифікації, форм власності, різновидів використання, управління, природоохоронних вимог та відповідальності за порушення законодавства.

понад двадцяти міжнародних конвенціях та двосторонніх угодах, які стосуються заходів з охорони навколишнього природного середовища. Міжнародні зобов'язання України, що стосуються використання природних ресурсів та забезпечення екологічної безпеки, випливають як з положень вже ратифікованих, так і тих, що перебувають на розгляді конвенцій та угод. Виконання зобов'язань передбачає приведення внутрішнього законодавства України у відповідність із нормами міжнародного права та урахування міжнародної практики при розробці нових законодавчих актів.

Отже, екологічне законодавство України є цілісною системою, яка складається з окремих законодавчих актів, що перебувають між собою у взаємодії.

ВИСНОВКИ

Результатом виконаної роботи є вирішення задачі побудови власної реалізації програмного продукту, спрямованого на захист від атак типу DDoS.

У процесі виконання роботи отримані наступні результати:

1. Проаналізовано нормативно-правову базу щодо забезпечення національної безпеки України в інформаційній сфері від DDoS-атак, розкрито сутність поняття DDoS-атак та їх видів, визначено механізми реалізації DDoS-атак та з’ясовано діючі методи захисту від них, висвітлено вплив DDoS-атак на веб-ресурси. Це стало основою для подальших досліджень у даній області.

2. Здійснено аналіз мови програмування та середовища розробки, що необхідні для ефективної розробки програмного продукту для протидії DDoS-атак. В результаті було виявлено, що Python є ідеальною мовою для майбутньої програми, забезпечуючи ефективність та масштабованість, до того ж середовище розробки PyCharm було вибрано за його функціональність, а для веб-інтерфейсу було обрано Flask та Tkinter за їх простоту, безпеку та можливість розширення.

3. На основі проведених досліджень ми здійснили проектування та розробку програмного продукту для ефективного захисту від DDoS-атак. Основна мета розробленої програми полягає у реалізації системи захисту від DDoS-атак та одночасно наданні можливості редагування та збереження списку IP-адрес. Тестування підтвердило успішну роботу захисної системи в умовах обмеження запитів та управління доступом до IP-адрес.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конституція України: Закон України від 28.06.1996 № 254к/96-ВР. URL: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96-%D0%B2%D1%80#Text (дата звернення 23.10.2023).

2. Про Стратегію національної безпеки України: Указ Президента України від 14.09.2020 №392/2020. URL: https://zakon.rada.gov.ua/laws/card/n0005525-20 (дата звернення 23.10.2023).

3. Про затвердження Положення про організаційно-технічну модель кіберзахисту: постанова Кабінету Міністрів України від 29.12.2021 № 1426. URL: https://zakon.rada.gov.ua/laws/show/1426-2021-%D0%BF#Text (дата звернення 23.10.2023).

4. Про затвердження Загальних вимог до кіберзахисту об'єктів критичної інфраструктури: постанова Кабінету Міністрів України від 19.06.2019 № 518. URL: https://zakon.rada.gov.ua/laws/show/518-2019-%D0%BF#Text (дата звернення 23.10.2023).

5. Про основні засади забезпечення кібербезпеки України: Закон України від 05.10.2017 № 2163-VIII. URL: https://zakon.rada.gov.ua/laws/show/2163-19#doc_info (дата звернення 23.10.2023).

6. Що таке DDoS-атака? URL: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-a-ddos-attack (дата звернення 24.10.2023).

7. Volumetric DDoS Attacks. URL: https://www.netscout.com/what-is-ddos/volumetric-attacks (дата звернення 24.10.2023).

8. Preventing Resource Exhaustion Cyberattacks: A Comprehensive Guide for IT Security Professionals. URL: https://dwomowale.medium.com/preventing-resource-exhaustion-cyberattacks-a-comprehensive-guide-for-it-security-professionals-a575ce59934 (дата звернення 24.10.2023).

9. What is an application layer attack? URL: https://www.nomios.com/resources/what-is-an-application-layer-attack/ (дата звернення 24.10.2023).

10. What are Bandwidth Attacks? URL: https://www.geeksforgeeks.org/what-are-bandwidth-attacks/ (дата звернення 24.10.2023).

11. What Is a DDoS Attack? URL: https://www.akamai.com/glossary/what-is-ddos (дата звернення 24.10.2023).

12. Hybrid Attack. URL: https://www.hypr.com/security-encyclopedia/hybrid-attack (дата звернення 24.10.2023).

13. What is a Volumetric DDoS Attack? URL: https://www.a10networks.com/glossary/what-is-a-volumetric-ddos-attack/ (дата звернення 25.10.2023).

14. CWE-400: Uncontrolled Resource Consumption. URL: https://cwe.mitre.org/data/definitions/400.html (дата звернення 25.10.2023).

15. Application layer DDoS attack. URL: https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/ (дата звернення 25.10.2023).

16. Bandwidth attacks. URL: https://www.usenix.org/legacy/publications/library/proceedings/sec01/full_papers/gil/gil_html/node3.html (дата звернення 25.10.2023).

17. What is a protocol DDoS attack? URL: https://www.rcrwireless.com/20230222/security/what-is-a-protocol-ddos-attack (дата звернення 25.10.2023).

18. Hybrid password attacks: How they work and how to stop them. URL: https://specopssoft.com/blog/hybrid-password-attacks/ (дата звернення 25.10.2023).

19. Як виправити помилку 503 Service Unavailable. URL: https://hostiq.ua/wiki/ukr/503-service-unavailable-error/ (дата звернення 26.10.2023).

20. The Damaging Impacts of DDoS Attacks. URL: https://www.corero.com/the-damaging-impacts-of-ddos-attacks/ (дата звернення 26.10.2023).

21. How much will a DDoS attack cost your small business? URL: https://www.techinsurance.com/resources/ddos-small-business-costs (дата звернення 26.10.2023).

22. The True Cost of a DDoS Attack: Protect Your Business with Proactive Measures. URL: https://www.linkedin.com/pulse/true-cost-ddos-attack-protect-your-business-proactive-ali-el-tom (дата звернення 26.10.2023).

23. Advantages of Python | Top Reasons to Learn Python. URL: https://www.simplilearn.com/why-learn-python-a-guide-to-unlock-your-python-career-article (дата звернення 30.10.2023).

24. Benefits of Python over Other Programming Languages. URL: https://www.invensis.net/blog/benefits-of-python-over-other-programming-languages (дата звернення 30.10.2023).

25. Python Language advantages and applications. URL: https://www.geeksforgeeks.org/python-language-advantages-applications/ (дата звернення 30.10.2023).

26. Advantages of Java. URL: https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-Advantages-Benefits-Fast-Performance-Simple-Open-Typed-Features-Streams (дата звернення 30.10.2023).

27. A Comprehensive Guide to C++: Advantages and Disadvantages. URL: https://pangea.ai/blog/languages/a-comprehensive-guide-to-c-advantages-and-disadvantages (дата звернення 30.10.2023).

28. Go programming language: utilities, features and advantages. URL: https://www.mytaskpanel.com/go-programming-language/ (дата звернення 30.10.2023).

29. Python’s Involvement in Advanced DDoS Attacks. URL: https://www.codewithc.com/pythons-involvement-in-advanced-ddos-attacks/ (дата звернення 30.10.2023).

30. PyCharm: all about the most popular Python IDE. URL: https://datascientest.com/en/pycharm-all-about-the-most-popular-python-ide (дата звернення 31.10.2023).

31. What is PyCharm? Features, Advantages & Disadvantages. URL: https://hackr.io/blog/what-is-pycharm (дата звернення 31.10.2023).

32. 20 Must-have plugins for PyCharm in 2022. URL: https://duckly.com/blog/best-plugins-for-pycharm-2022/ (дата звернення 31.10.2023).

33. Python Flask: pros and cons. URL: https://dev.to/detimo/python-flask-pros-and-cons-1mlo (дата звернення 01.11.2023).

34. Pros and Cons of Tkinter. URL: https://www.oreilly.com/library/view/python-programming-on/1565926218/ch20s01s02.html (дата звернення 01.11.2023).

35. Діаграма пакетів. URL: https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D1%96%D0%B2 (дата звернення 06.11.2023).

36. Архітектура програмного забезпечення: все що треба знати. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення 07.11.2023).

37. Діаграма розгортання. URL: https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D1%80%D0%BE%D0%B7%D0%B3%D0%BE%D1%80%D1%82%D0%B0%D0%BD%D0%BD%D1%8F (дата звернення 08.11.2023).

38. Тестування програмного забезпечення. URL: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата звернення 09.11.2023).

39. СТАТИЧНА І ДИНАМІЧНА МЕТОДИКИ ТЕСТУВАННЯ. URL: https://training.qatestlab.com/blog/technical-articles/static-and-dynamic-testing-methods/ (дата звернення 09.11.2023).

40. Дослідницьке тестування – глибоке занурення в типи, процеси, підходи, інструменти, фреймворки та багато іншого! URL: https://www.zaptest.com /uk/%D0%B4%D0%BE%D1%81%D0%BB%D1%96%D0%B4%D0%BD%D0%B8 %D1%86%D1%8C%D0%BA%D0%B5-%D1%82%D0%B5%D1%81%D1%82% D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B3%D0%BB%D0 %B8%D0%B1%D0%BE%D0%BA%D0%B5-%D0%B7%D0%B0%D0%BD (дата звернення 09.11.2023).

41. Про охорону навколишнього природного середовища: Закон України від 25.06.1991 № 1264-XII. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення 12.12.2023).

42. Про оцінку впливу на довкілля: Закон України від 23.05.2017 № 2059-VIII. URL: https://zakon.rada.gov.ua/laws/show/2059-19#Text (дата звернення 12.12.2023).

43. Основи законодавства України про охорону здоров'я: Закон України від 19.11.1992 № 2801-XII. URL: https://zakon.rada.gov.ua/laws/show/2801-12#Text (дата звернення 12.12.2023).

44. Про підприємництво: Закон України від 07.02.1991 № 698-XII. URL: https://zakon.rada.gov.ua/laws/show/698-12#Text (дата звернення 12.12.2023).

45. Про інформацію: Закон України від 02.10.1992 № 2657-XII. URL: https://zakon.rada.gov.ua/laws/show/2657-12#Text (дата звернення 12.12.2023).

46. Податковий кодекс України: від 02.12.2010 № 2755-VI. URL: https://zakon.rada.gov.ua/laws/show/2755-17#Text (дата звернення 12.12.2023).

47. Кодекс України про адміністративні правопорушення: від 07.12.1984 № 8073-X. URL: https://zakon.rada.gov.ua/laws/show/2755-17#Text (дата звернення 12.12.2023).

48. Кримінальний кодекс України: від 05.04.2001 № 2341-III. URL: https://zakon.rada.gov.ua/laws/show/2341-14#Text (дата звернення 12.12.2023).

49. Земельний кодекс України: від 25.10.2001 № 2768-III. URL: https://zakon.rada.gov.ua/laws/show/2768-14#Text (дата звернення 12.12.2023).

50. Водний кодекс України: від 06.06.1995 № 213/95-ВР. URL: https://zakon.rada.gov.ua/laws/show/213/95-%D0%B2%D1%80#Text (дата звернення 12.12.2023).

51. Лісовий кодекс України: від 21.01.1994 № 3852-XII. URL: https://zakon.rada.gov.ua/laws/show/3852-12#Text (дата звернення 12.12.2023).

52. Про надра: Кодекс України від 27.07.1994 № 132/94-ВР. URL: https://zakon.rada.gov.ua/laws/show/132/94-%D0%B2%D1%80#Text (дата звернення 12.12.2023).

53. Гірничий закон України: Закон України від 06.10.1999 № 1127-XIV. URL: https://zakon.rada.gov.ua/laws/show/1127-14#Text (дата звернення 12.12.2023).

54. Про тваринний світ: Закон України від 13.12.2001 № 2894-III. URL: https://zakon.rada.gov.ua/laws/show/2894-14#Text (дата звернення 12.12.2023).

55. Про охорону атмосферного повітря: Закон України від 16.10.1992 № 2707-XII. URL: https://zakon.rada.gov.ua/laws/show/2707-12#Text (дата звернення 12.12.2023).

56. Про природно-заповідний фонд України: Закон України від 16.06.1992 № 2456-XII. URL: https://zakon.rada.gov.ua/laws/show/2456-12#Text (дата звернення 12.12.2023).

57. Про виключну (морську) економічну зону України: Закон України від 16.05.1995 № 162/95-ВР. URL: https://zakon.rada.gov.ua/laws/show/162/95-%D0%B2%D1%80#Text (дата звернення 12.12.2023).

58. Про Червону книгу України: Закон України від 07.02.2002 № 3055-III. URL: https://zakon.rada.gov.ua/laws/show/3055-14#Text (дата звернення 12.12.2023).

Додаток А

Код програмного продукту

```
from flask import Flask, render_template, request
from flask_limiter import Limiter
import threading
import tkinter as tk
from tkinter import filedialog
from tkinter import messagebox

app = Flask(__name__)
limiter = Limiter(app)
limiter.init_app(app)

class IpHandler:
    FILE_PATH = "B://Dyploma/iplist.txt"

    @staticmethod
    def write_to_file(ip):
        try:
            with open(IpHandler.FILE_PATH, 'a') as file:
                file.write(ip + "\n")
            print(f"IP {ip} added to the file.")
        except Exception as e:
            print(f"Error writing to file: {e}")

    @staticmethod
    def get_ip_list():
        return FileHandler.open_file()
```

```
    @staticmethod
    def add_ip_address(ip, ip_addresses):
        print(f"Processing IP: {ip}")

        ip_addresses.append(ip)
        print(f"IP {ip} added to the list.")

        # Додати IP в файл незалежно від лімітів
        IpHandler.write_to_file(ip)

        if len(ip_addresses) > 10000:
            ip_addresses.pop(0)

class FileHandler:
    FILE_PATH = "B://Dyploma/iplist.txt"

    @staticmethod
    def open_file():
        try:
            with open(FileHandler.FILE_PATH, 'r') as file:
                content = file.readlines()
            return content
        except FileNotFoundError:
            return []

    @staticmethod
    def save_file(content):
        try:
            with open(FileHandler.FILE_PATH, 'w') as file:
                file.writelines(content)
```

```
        except Exception as e:
            print(f"Error saving file: {e}")


@app.route('/')
@limiter.limit("5 per second")
def index():
    global temp_limit_counter
    ip_address = request.headers.get('X-Forwarded-For', request.remote_addr)

    # Перевірка, чи IP вже перевищив тимчасовий ліміт
    if ip_address in ip_temp_limits and ip_temp_limits[ip_address] >=
TEMP_LIMIT_THRESHOLD:
        # Додати IP в файл, якщо тимчасовий ліміт досягнуто
        IpHandler.add_ip_address(ip_address, ip_addresses)
        # Видалити IP з тимчасового обмеження
        ip_temp_limits.pop(ip_address)
        print(f"IP {ip_address} added to the file due to reaching temporary limit.")

    # Отримання IP-адрес з iplist.txt
    ip_list = FileHandler.get_ip_list()

    return render_template('index.html', ip_addresses=ip_addresses,
ip_list=ip_list)

class TextEditorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Text Editor with DDoS Protection")

        self.ip_handler = IpHandler()
```

```
        self.file_handler = FileHandler()

        left_frame = tk.Frame(root)
        left_frame.pack(side="left", padx=10)

        open_button = tk.Button(left_frame, text="Open IpList",
command=self.open_file)
        open_button.grid(row=0, column=0, pady=5)

        save_button = tk.Button(left_frame, text="Save IpList",
command=self.save_file)
        save_button.grid(row=1, column=0, pady=5)

        self.search_entry = tk.Entry(left_frame)
        self.search_entry.grid(row=2, column=0, pady=5)

        add_button = tk.Button(left_frame, text="Add ip",
command=self.add_text)
        add_button.grid(row=3, column=0, pady=5)

        delete_button = tk.Button(left_frame, text="Delete ip",
command=self.delete_text)
        delete_button.grid(row=4, column=0, pady=5)

        right_frame = tk.Frame(root)
        right_frame.pack(side="right", padx=10)

        self.text_editor = tk.Text(right_frame, wrap="word", width=40, height=10)
        self.text_editor.pack(pady=10)
```

```
def open_file(self):
    content = self.file_handler.open_file()
    self.display_content(content)

def save_file(self):
    content = self.text_editor.get(1.0, tk.END)
    self.file_handler.save_file(content)

def add_text(self):
    new_text = self.search_entry.get()
    self.text_editor.insert(tk.END, new_text + "\n")

def delete_text(self):
    target_text = self.search_entry.get()
    text_content = self.text_editor.get(1.0, tk.END)

    lines = text_content.split("\n")
    updated_lines = [line for line in lines if line.strip() != target_text]

    updated_content = "\n".join(updated_lines)

    self.text_editor.delete(1.0, tk.END)
    self.text_editor.insert(tk.END, updated_content)
    self.remove_empty_lines()
    self.text_editor.insert(tk.END, "\n")

def remove_empty_lines(self):
    content = self.text_editor.get(1.0, tk.END)
    lines = content.split("\n")
    non_empty_lines = [line for line in lines if line.strip()]
```

```
        updated_content = "\n".join(non_empty_lines)
        self.text_editor.delete(1.0, tk.END)
        self.text_editor.insert(tk.END, updated_content)

    def display_content(self, content):
        self.text_editor.delete(1.0, tk.END)
        self.text_editor.insert(tk.END, "\n".join(content) + "\n")

def run_flask_in_thread():
    app.run(debug=False, host='0.0.0.0')

def toggle_protection():
    global temp_limit_counter
    if limiter.enabled:
        limiter.enabled = False
        status_label.config(text="Protection Disabled")
    else:
        limiter.enabled = True
        status_label.config(text="Protection Enabled")
        # Скидання лічильника тимчасових обмежень при включенні захисту
        temp_limit_counter = 0

def on_closing():
    limiter.enabled = False
    root.destroy()

root = tk.Tk()
root.title("DDoS Protection and Text Editor")

status_label = tk.Label(root, text="Protection Enabled", font=("Helvetica", 12))
```

```
status_label.pack(pady=10)

toggle_button = tk.Button(root, text="Toggle Protection",
command=toggle_protection)
toggle_button.pack(pady=10)

ip_addresses = []
ip_temp_limits = {}
temp_limit_counter = 0
TEMP_LIMIT_THRESHOLD = 6

flask_thread = threading.Thread(target=run_flask_in_thread)
flask_thread.daemon = True
flask_thread.start()

text_editor_app = TextEditorApp(root)

try:
    root.mainloop()
except KeyboardInterrupt:
    sys.exit(0)
```