Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
в.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: Програмний модуль забезпечення захисту вебзастосунків від DDoS-атак

Виконавець: студент IV курсу, групи КБ-42

_____ **Антон Шпильовий**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Андрій Фесенко	

Нормоконтроль	Андрій Бігдан	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
в.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-42** (група) **Шпильовому Антону Миколайовичу** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Програмний модуль забезпечення захисту вебзастосунків від DDoS-атак

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Види DDoS-атак, методи захисту від DDoS-атак

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з принципами захисту від DDoS-атак, розглянути та проаналізувати основні методи захисту, розробити рекомендації щодо розробки програмного модуля захисту від атак, реалізувати алгоритм ідентифікації атак, розробити механізм реагування на атаку.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблений програмний модуль для забезпечення захисту від DDoS-атак, а також графічний інтерфейс до нього.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Андрій Фесенко (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Антон Шпильовий (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 24.10.2023	*виконано*
2	Аналіз літератури	29.01.2023 – 11.02.2023	*виконано*
3	Обґрунтування вибору рішення	12.02.2023 – 15.02.2023	*виконано*
4	Вивчення основних понять, історії та природи DDoS-атак	16.02.2023 – 04.03.2023	*виконано*
5	Аналіз принципу дії DDoS-атак	05.03.2023 – 21.03.2023	*виконано*
6	Дослідження методів захисту від DDoS-атак	22.03.2023 – 08.04.2023	*виконано*
7	Розробка програмного забезпечення для захисту вебзастосунків від DDoS-атак	09.04.2023 – 10.05.2023	*виконано*
8	Оформлення пояснювальної записки	11.05.2023 – 27.05.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	28.05.2023 – 12.06.2023	*виконано*

Завдання видав _____ (підпис) Андрій Фесенко (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Антон Шпильовий (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна містить в собі вступ, три розділи, загальні висновки, список використаних джерел та додатки. Основний текст займає 64 сторінки. Даний контент включає в себе зміст, вступ, три розділи дипломної роботи, висновки та список джерел. Також в роботі є три додатки, два з яких – лістинг програмного коду, а інший містить список опублікованих праць. У пояснювальній записці даної роботи міститься 19 рисунків та 1 таблиця. Список використаних джерел містить 30 найменувань та займає 3 сторінки.

Метою роботи є розробка програмного модуля для захисту вебзастосунків від DDoS-атак.

Об'єктом дослідження є процес захисту вебзастосунків від DDoS-атак.

Предметом дослідження є система виявлення і протидії DDoS-атакам.

Методи дослідження включають структурний аналіз, порівняння та формалізація.

Практична цінність роботи полягає в методах обробки http-запитів до вебзастусунку.

Ключові слова: захист від DDoS-атак, розробка програмного забезпечення, мережевий екран, алгоритм ідентифікації DDoS-атаки, захист за допомогою PHP, вебсокет з'єднання, механізм блокування IP-адреси, інтерфейс для спостереження, аналіз видів DDoS-атак.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

URL	–	Uniform Resource Locator
API	–	Application Programming Interface
(D)DoS	–	(Distributed) Denial-of-Service
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
ICMP	–	Internet Control Message Protocol
DNS	–	Domain Name System
HTTP	–	HyperText Transfer Protocol
NTP	–	Network Time Protocol
WS	–	WebSocket
CDN	–	Content Delivery Network
ОС	–	Операційна Система
TOR	–	The Onion Router
IPS	–	Intrusion Prevention System
IDS	–	Intrusion Detection System
VPS	–	Virtual Private Server
DOM	–	Document Object Model

ЗМІСТ

ВСТУП

Актуальність даної роботи визначається тією обставиною, що на даний момент DDoS-атаки є дуже поширеним явищем. Це призводить до колосальних збитків серед компаній.

DDoS-атака - це тип кібератаки, що має на меті заблокувати доступ до мережевих ресурсів шляхом їх перевантаження. Це може бути виконано шляхом надсилання великої кількості запитів до сервера, що змушує його працювати з підвищеною навантаженістю, що може призвести до збою в роботі або повільної реакції. DDoS атаки "розподілені", оскільки вони використовують багато різних комп'ютерів або інших пристроїв, щоб виконувати атаку. Це робить їх складнішими для протидії, оскільки трафік, який вони генерують, приходить з багатьох різних джерел. Цілі DDoS атаки можуть варіюватись, включаючи відповідь на конкурентний бізнес, політичні мотиви, або просто завдання шкоди. Вони можуть бути особливо шкідливими для бізнесу, оскільки вони можуть зупинити роботу вебсайтів або інших онлайн-сервісів, викликаючи втрату продажів та пошкодження репутації. Принцип здійснення DDoS-атаки зображено на рисунку 1.1.

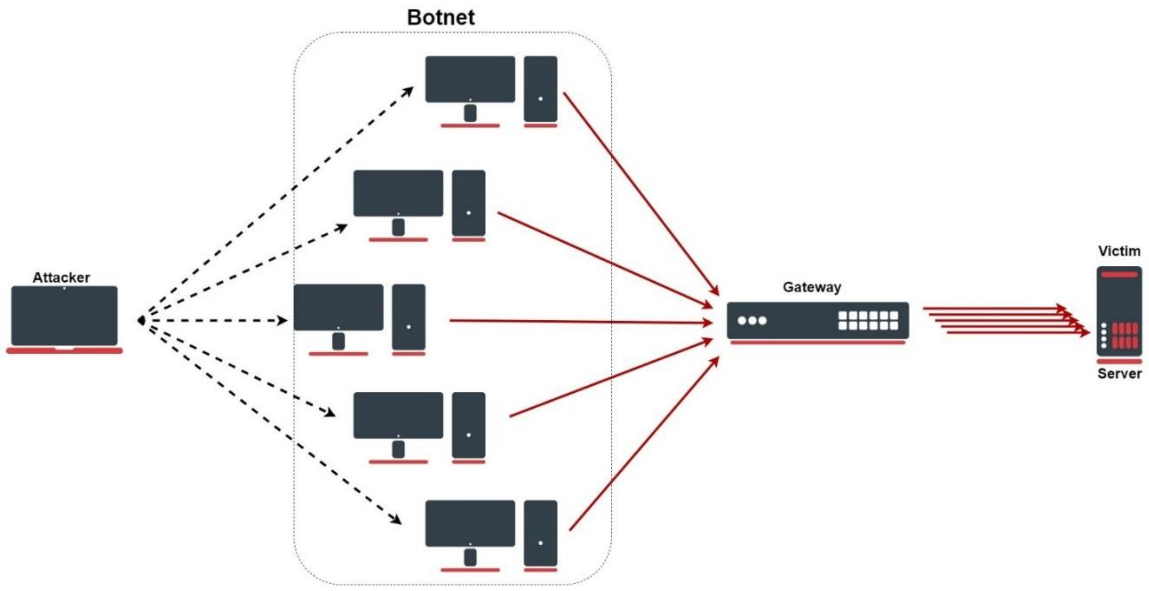

Рисунок 1.1 – Принцип здійснення DDoS-атаки

Перша відома DDoS-атака сталася в 1999 році і називалася "атака залпом" (smurf attack). Ця атака була результатом використання специфічного експлойта в протоколі Internet Control Message Protocol (ICMP), який використовується для відправки повідомлень про помилки та діагностики в мережах IP. Атака залпом використовувала підроблені IP-адреси, щоб надсилати ICMP-запити (Echo Request) до широкомовних адрес (broadcast addresses) в мережі. У відповідь на кожен запит, всі пристрої в цій підмережі надсилали відповідь (Echo Reply) до підробленої адреси, перенавантажуючи обрану цільову систему або мережу. Ця атака засновувалася на використанні широкомовного розсилання (broadcasting) та вразливості протоколу ICMP, яка дозволяла зловмисникам замаскувати свою справжню адресу IP і відправляти запити від іншого джерела. Це дозволяло створювати залпову активність і перенавантажувати системи. Дана атака викликала значні проблеми інтернет-провайдерам і веб-сайтам того часу. Вона показала, що мережева інфраструктура і протоколи можуть бути використані для спрямованих атак на інші системи. Після першої атаки залпом з'явилося багато інших форм DDoS-атак, які використовують різні методи і засоби для перевантаження цільових систем та мереж.

DDoS-атаки мають досить довгу історію, яка починається в кінці 1990-х років. Ось короткий огляд етапів розвитку DDoS-атак:

1. Етап піонерів (кінець 1990-х - початок 2000-х): перші DDoS-атаки стали можливими завдяки використанню зловмисниками ботнетів, що складалися з компрометованих комп'ютерів. Ці атаки зазвичай спрямовувалися на відомі веб-сайти, зловживаючи надмірними запитами, що призводили до перевантаження серверів і відмови обслуговування.

2. Етап комерціалізації (середина 2000-х): на цьому етапі DDoS-атаки стали комерційними послугами, доступними для замовлення. Зловмисники надавали послуги зі спрямованою службою, пропонуючи атаки на замовлення. Це призвело до збільшення числа атак та зростання їх розмаху.

3. Етап ботнетів (2000-і - сьогодні): найбільш сучасний етап розвитку DDoS-атак пов'язаний з використанням ботнетів. Ботнет – це мережа комп'ютерів, що складаються з компрометованих систем, які можуть бути використані для здійснення

масштабних DDoS-атак. Зловмисники набирають доступ до цих ботнетів і використовують їх для нападів на обрані цілі.

4. Етап розвитку захисту (середина 2000-х – сьогодні): з виникненням загрози DDoS-атак з'явилися рішення для їх захисту. Розробники стали пропонувати різноманітні системи та послуги, спрямовані на виявлення, блокування та мінімізацію впливу DDoS-атак. Це включає в себе використання спеціалізованих апаратних пристроїв, софтверних рішень, хмарних платформ та інших методів захисту.

DDoS-атаки залишаються серйозною загрозою в сучасному світі, і вони продовжують еволюціонувати разом зі зростанням обсягів мереж та доступних ресурсів для зловмисників. Розвиток захисту та вдосконалення технологій захисту є невід'ємною складовою боротьби з цими атаками.

Сьогодні тисячі спеціалістів сумлінно працюють над збільшенням рівня захисту від DDoS-атак. Ціль захисту від DDoS-атак полягає в забезпеченні доступності, надійності та стабільності мережі та інтернет-сервісів, уникненні втрати доходів та психологічної шкоди, яку можуть заподіяти такі атаки.

Основні цілі захисту від DDoS-атак включають:

- Забезпечення доступності сервісів: Основною метою DDoS-атак призупинення або зниження доступності веб-сайтів, онлайн-платформ, мережевих служб та інших ресурсів. Захист від DDoS-атак спрямований на забезпечення нормального функціонування цих ресурсів та відновлення їх роботи після атаки.
- Запобігання фінансовим збиткам: DDoS-атаки можуть мати серйозні фінансові наслідки для організацій, особливо тих, що залежать від онлайн-торгівлі, рекламного доходу або інших електронних послуг. Захист від DDoS-атак спрямований на забезпечення безперебійної роботи бізнесу та запобігання фінансовим втратам.
- Захист репутації: DDoS-атаки можуть негативно вплинути на репутацію організації, особливо якщо вона не здатна забезпечити доступність своїх послуг користувачам. Захист від DDoS-атак спрямований на збереження доброї репутації, забезпечуючи надійність та стабільність обслуговування клієнтів.

- Забезпечення безпеки: деякі DDoS-атаки можуть бути використані як засіб відволікання або маскування для інших кібератак, наприклад, для злому систем або викрадення даних. Захист від DDoS-атак також має на меті забезпечення безпеки мережі та захист від інших загроз кібербезпеки.

Метою роботи є розробка програмного модуля для захисту вебзастосунків від DDoS-атак.

Для досягнення зазначеної мети дипломної роботи поставлено наступні **завдання**:

- провести аналіз видів DDoS-атак;
- дослідити методи захисту від DDoS-атак;
- виділити рекомендації необхідні для розробки програмного модуля;
- розробити програмну реалізацію модуля захисту від DDoS-атак;
- описати програмну реалізацію.

Об'єктом дослідження є процес захисту вебзастосунків від DDoS-атак.

Предметом дослідження є система виявлення і протидії DDoS-атакам.

Методи дослідження: структурний аналіз, порівняння, формалізація.

Практична цінність роботи полягає в наступному:

- розробка архітектури системи захисту від DDoS-атак;
- методи обробки http-запитів до вебзастусунку;
- реалізація програмного модуля захисту від DDoS-атак на мові програмування PHP для серверів на базі ОС Linux.

РОЗДІЛ 1
АНАЛІЗ ВИДІВ ТА МЕТОДІВ ЗАХИСТУ ВІД DDOS-АТАК

1.1 Атаки на рівні мережі

DDoS-атаки на рівні мережі, також відомі як атаки на пропускну здатність мережі (network-layer DDoS attacks), спрямовані на перевантаження мережевої інфраструктури, щоб знизити або заблокувати доступ до цільових ресурсів. Ці атаки націлені на використання вразливостей протоколів мережі або споживання обмежених ресурсів, таких як пропускна здатність, CPU або пам'ять мережевих пристроїв.

Основні типи DDoS-атак на рівні мережі включають:

- Атаки на пропускну здатність: ці атаки спрямовані на перевантаження пропускної здатності мережевих каналів. Зловмисники генерують великі обсяги трафіку, які перевищують пропускну здатність мережі, що призводить до відмови в обслуговуванні для легітимного трафіку.
- SYN флуд: ця атака використовує вразливість у TCP-протоколі, викликаючи відкриття багатьох неповних з'єднань (half-open connections), залишаючи їх незавершеними. Це може призвести до перевантаження ресурсів сервера та відмови в обслуговуванні.
- ICMP флуд: в цій атаки зловмисники використовують ICMP-пакети (ping-запити), щоб надсилати великий обсяг запитів до цільових мережевих пристроїв, що може спричинити перевантаження їх ресурсів та відмову в обслуговуванні.
- UDP флуд: ця атака спрямована на перевантаження мережі за допомогою великого обсягу UDP-пакетів. Оскільки UDP є безз'єднаним протоколом, немає необхідності встановлювати з'єднання, що дозволяє зловмисникам відправляти велику кількість пакетів на цільову мережу.

- DNS ампліфікація: ця атака використовує вразливості в DNS-серверах для посилення атаки. Зловмисники надсилають запити до вразливих DNS-серверів з підробленими даними, що призводить до надмірної відповіді DNS-сервера на адресу жертви, перевантажуючи її мережу.

Захист від DDoS-атак на рівні мережі включає використання мережевих пристроїв, які можуть виявляти та фільтрувати небажаний трафік, встановлення пропускних обмежень, використання IDS/IPS систем для виявлення та блокування атак, а також використання розподілених систем захисту (наприклад, Cloudflare), які можуть розсіювати трафік та фільтрувати шкідливі запити.

1.2 Атаки на рівні транспортного протоколу

DDoS-атаки на рівні транспортного протоколу, також відомі як атаки на транспортний рівень, спрямовані на перевантаження протоколів транспортного рівня, зокрема TCP або UDP, для завади нормальному функціонуванню цільових сервісів.

Основні типи DDoS-атак на рівні транспортного протоколу включають:

- SYN флуд: цей тип атаки використовує вразливість у TCP-протоколі. Зловмисники надсилають багато запитів на встановлення з'єднання (SYN), але не завершують їх, залишаючи відкриті недовстановлені з'єднання (half-open connections). Це може призвести до вичерпання ресурсів сервера і відмови в обслуговуванні легітимного трафіку.
- UDP флуд: ця атака використовує UDP-пакети для перевантаження мережі та цільових ресурсів. Зловмисники надсилають велику кількість UDP-пакетів до цільової системи, що може призвести до перевантаження її ресурсів та відмови в обслуговуванні.
- TCP ACK флуд: ця атака використовує велику кількість підроблених ACK-пакетів (Acknowledgment) для навантаження сервера. Зловмисники відправляють підроблені ACK-пакети без фактичного відкриття з'єднання, що може спричинити перевантаження сервера та відмову в обслуговуванні.

- RST флуд: ця атака використовує підроблені RST-пакети (Reset) для припинення активних з'єднань. Зловмисники надсилають підроблені RST-пакети до цільової системи, що може спричинити зниження якості обслуговування або відмову в обслуговуванні.

Захист від DDoS-атак на рівні транспортного протоколу може включати використання мережевих пристроїв, які здатні виявляти та фільтрувати небажаний трафік на основі правил, встановлення відповідних пропускних обмежень, використання IPS/IDS систем для виявлення та блокування атак, а також розгортання механізмів, таких як TCP, SYN, Cookies, які можуть допомогти уникнути перевантаження внаслідок SYN флуду.

1.3 Атаки на рівні застосунків

DDoS-атаки на рівні застосунків, також відомі як атаки на рівень додатків (application-layer DDoS attacks), спрямовані на перевантаження конкретного додатку або сервісу, шляхом використання вразливостей або обтяжуючих запитів.

Основні види DDoS-атак на рівні застосунків включають:

- HTTP флуд: цей тип атаки спрямований на перевантаження веб-серверів шляхом надсилання великої кількості запитів HTTP. Зловмисники можуть використовувати ботнети або інші автоматизовані методи для надсилання великого обсягу запитів, перевантажуючи ресурси сервера та знижуючи доступність для легітимного трафіку.
- DNS ампліфікація: ця атака використовує вразливості в DNS-серверах для посилення атаки. Зловмисники надсилають запити до вразливих DNS-серверів з підробленими даними, що призводить до надмірної відповіді DNS-сервера на адресу жертви, перевантажуючи її мережу.
- SQL ін'єкція: ця атака спрямована на вразливості веб-додатків, які використовують SQL-запити до баз даних. Зловмисники вставляють шкідливі SQL-

запити в вхідні поля, що може призвести до витоку даних або перевантаження бази даних, спричиняючи відмову в обслуговуванні.

- UDP флуд: ця атака використовує UDP-пакети для перевантаження додатків або сервісів. Зловмисники надсилають велику кількість UDP-пакетів до цільової системи, що може призвести до перевантаження ресурсів та відмови в обслуговуванні.

Захист від DDoS-атак на рівні застосунків включає використання механізмів фільтрації трафіку, виявлення та блокування шкідливих запитів, моніторинг активності та аналіз логів для виявлення несправедливого трафіку, а також використання веб-фаєрволів та систем захисту, які можуть виявляти та блокувати аномальний або зловмисний трафік.

1.4 Ампліфікаційні атаки

Ампліфікаційні DDoS-атаки (amplification DDoS attacks) – це тип атак, в яких зловмисники використовують вразливості у деяких протоколах або сервісах для збільшення обсягу трафіку, який надсилається до цільової системи. Це дає можливість зловмисникам здійснювати атаки з великим обсягом трафіку, використовуючи відносно невелику вихідну пропускну здатність.

Процес ампліфікаційної DDoS-атаки включає наступні кроки:

1. Зловмисники знаходять вразливості або використовують відкриті ресурси в різних протоколах або сервісах.
2. Зловмисники використовують ці вразливості, щоб надсилати великі обсяги запитів до серверів, які мають бути використані як засоби ампліфікації. Ці запити зазвичай є короткими, але їх відповіді можуть бути значно більшими за розміром.
3. Зловмисники підробляють адресу IP атакуючого, щоб відправляти запити до цільової системи, і відповіді від серверів-ампліфікаторів направляються до цільової IP-адреси.

4. Це призводить до перевантаження ресурсів цільової системи через надмірний трафік, який генерується внаслідок ампліфікаційного ефекту.

Ампліфікаційні DDoS-атаки можуть використовувати різні протоколи та сервіси, такі як:

- DNS (Domain Name System): зловмисники використовують вразливості в DNS-серверах, щоб надсилати запити з підробленими даними, що спричиняють надмірну відповідь DNS-сервера на адресу жертви.
- NTP (Network Time Protocol): зловмисники використовують вразливості в NTP-серверах, щоб надсилати запити з підробленими даними, що викликають надмірну відповідь NTP-сервера, яка направляється до жертви.
- SNMP (Simple Network Management Protocol): зловмисники використовують вразливості в SNMP-серверах, щоб надсилати запити з підробленими даними, що викликають надмірну відповідь SNMP-сервера, яка направляється до жертви.
- Memcached: зловмисники використовують вразливості в Memcached-серверах, щоб надсилати запити з підробленими даними, що викликають надмірну відповідь Memcached-сервера, яка направляється до жертви.

Для захисту від ампліфікаційних DDoS-атак необхідно використовувати відповідні заходи безпеки, такі як фільтрація або блокування небажаного трафіку на вузлах мережі, налаштування серверів та сервісів для виявлення та обмеження ампліфікаційного трафіку, а також моніторинг мережі та вчасне реагування на випадки підозрілого або аномального трафіку.

Варто пам'ятати, що немає повного захисту від DDoS-атак, але застосування цих заходів може суттєво зменшити ризик і вплив таких атак на вашу систему. Рекомендується постійно оновлювати та переглядати заходи захисту для виявлення та вирішення нових загроз.

1.5 Синхронізовані атаки

Синхронізовані DDoS-атаки (synchronized DDoS attacks) є формою атак, в яких зловмисники координують дії і направляють велику кількість шкідливого трафіку до цільової системи одночасно з різних джерел. Це робить атаку більш ефективною і складнішою для виявлення та протидії.

Основна мета синхронізованих DDoS-атак полягає у перевантаженні ресурсів цільової системи, таких як мережеві канали, сервери або інфраструктура, шляхом одночасної надсилання великої кількості запитів або трафіку з багатьох джерел. Це може призводити до відмови в обслуговуванні або значного зниження продуктивності цільової системи, що призводить до недоступності для законних користувачів.

Синхронізовані DDoS-атаки можуть бути здійснені з використанням ботнетів, які складаються з компрометованих комп'ютерів і пристроїв, або шляхом зловживання вразливостей протоколів, що дозволяють зловмисникам використовувати ампліфікаційні техніки для збільшення обсягу трафіку.

Для захисту від синхронізованих DDoS-атак рекомендуються такі заходи:

- Моніторинг мережі та виявлення аномального трафіку, що може свідчити про синхронізовану атаку.
- Використання захисних систем, таких як файрволи, IPS (інтрузійні системи запобігання) та WAF (веб-фаєрволи), для фільтрації та блокування шкідливого трафіку.
- Розподілення трафіку за допомогою CDN (мережа доправлення контенту) або аналогічних технологій, що дозволяють розподіляти навантаження між різними серверами та мережевими точками присутності.
- Застосування систем обмеження швидкості або капчі, які можуть відсікати надмірний трафік від потенційно небезпечних джерел.
- Готовність до відновлення роботи після атаки шляхом резервного копіювання даних та розробки планів надійності та відновлення системи.

Важливо зауважити, що захист від синхронізованих DDoS-атак потребує комплексного підходу, який включає технічні, мережеві та процедурні заходи безпеки.

1.6 Мережевий екран і фільтрація трафіку

Мережевий екран і фільтрація трафіку є ефективними засобами захисту від DDoS-атак на рівні мережі. Вони дозволяють ідентифікувати і блокувати шкідливий трафік, спрямований на цільову систему, і забезпечувати безпеку мережі.

Мережевий екран (firewall) – це система, яка контролює трафік, що проходить через мережу. Він дозволяє налаштовувати правила фільтрації, які визначають, який трафік дозволений або заборонений на підставі різних параметрів, таких як IP-адреси, порти, протоколи тощо. Для захисту від DDoS-атак мережевий екран може бути налаштований для блокування аномального або надмірного трафіку, який може бути ознакою атаки.

Фільтрація трафіку - це процес відсіювання небажаного або шкідливого трафіку з мережі. Це може бути досягнуто за допомогою різних технік, таких як IP-фільтрація, фільтрація за портами, фільтрація за протоколами тощо. Фільтри можуть бути встановлені на різних рівнях мережевого стеку для блокування шкідливого трафіку досягає цільової системи.

Основні переваги мережевого екрана і фільтрації трафіку для захисту від DDoS-атак:

- Виявлення та блокування шкідливого трафіку: мережевий екран може аналізувати трафік, що протікає через мережу, і виявляти аномальний або шкідливий трафік, який може бути пов'язаний з DDoS-атаками. Фільтрація трафіку дозволяє відсікати небажаний трафік на основі певних параметрів.
- Зменшення навантаження на мережу: блокування шкідливого трафіку на рівні мережі дозволяє зменшити навантаження на мережеві ресурси, такі як мережеві

канали і комутатори, тим самим запобігаючи перевантаженню системи під час DDoS-атак.

- Гнучкість налаштувань: мережеві екрани та фільтри трафіку надають можливість гнучко налаштувати правила фільтрації, що дозволяє адаптувати їх до конкретних потреб і характеристик мережі.
- Зменшення ризику проникнення в шкідливий трафік: використання мережевого екрана та фільтрації трафіку допомагає запобігати проникненню шкідливого трафіку до системи, зменшуючи загрозу безпеці та ризик компрометації даних.

Застосування мережевого екрана і фільтрації трафіку в поєднанні з іншими заходами безпеки, такими як мережевий моніторинг, захист від ампліфікаційних атак та використання CDN (мережа доправлення контенту), допомагає підвищити рівень захисту від DDoS-атак на рівні мережі.

1.7 Захист на рівні мережі

Захист на рівні мережі від DDoS-атак є важливою складовою стратегії безпеки і дозволяє запобігти атакам на мережевий ресурс і забезпечити стабільну роботу системи. Основні методи захисту на рівні мережі від DDoS-атак включають:

- Firewall: використання мережевого екрана (firewall) дозволяє контролювати трафік, що проходить через мережу, і блокувати небажаний або шкідливий трафік. Встановлення правил фільтрації на мережевому екрані дозволяє ідентифікувати та блокувати DDoS-трафік, спрямований на цільову систему.
- Intrusion Detection and Prevention Systems (IDS/IPS): системи виявлення і запобігання вторгнень в мережу допомагають виявляти аномальний або шкідливий трафік, пов'язаний з DDoS-атаками, і приймати відповідні заходи для його блокування.
- Traffic Scrubbing: Технології очищення трафіку (traffic scrubbing) використовуються для відфільтрування шкідливого трафіку, що надходить до мережі.

Ці технології використовують спеціалізоване обладнання та аналізують пакети трафіку, щоб виділити шкідливі дані та блокувати їх.

- Rate Limiting: обмеження швидкості передачі даних (rate limiting) може бути застосоване для контролю надмірного трафіку, що надходить до системи. Це дозволяє зменшити навантаження на мережу та запобігти перевантаженню ресурсів.
- BGP Flowspec - це протокол, який дозволяє операторам мереж визначати та розповсюджувати правила фільтрації для контролю трафіку. Використання BGP Flowspec дозволяє оперативно реагувати на DDoS-атаки і блокувати шкідливий трафік на рівні мережі.

Ці методи та технології можуть бути використані окремо або в комбінації для підвищення рівня захисту на рівні мережі від DDoS-атак. Ефективний захист вимагає постійного моніторингу мережі, виявлення аномалій та швидкої реакції на потенційні атаки.

1.8 Захист на рівні застосунків

Захист від DDoS-атак на рівні застосунків є одним з етапів повноцінної стратегії захисту. Цей рівень захисту спрямований на ідентифікацію і фільтрацію шкідливого трафіку на рівні самого застосунку. Основні методи та підходи до захисту на рівні застосунків від DDoS-атак включають:

- Фільтрація запитів: встановлення правил фільтрації та перевірка вхідних запитів на відповідність цим правилам. Запити, які не відповідають правилам, можуть бути блоковані або оброблені окремим чином, щоб запобігти перевантаженню сервера або погіршенню доступності.
- Кешування контенту: використання технологій кешування динамічного або статичного контенту може зменшити навантаження на сервер і покращити доступність застосунку під час DDoS-атаки. Кешування дозволяє відповідати на запити користувачів без звернення до сервера, що допомагає знизити навантаження на ресурси.

- Захист від зламу: використання механізмів захисту від зламу допомагає запобігти використанню вразливостей застосунку для запуску DDoS-атак або отримання несанкціонованого доступу. Це може включати використання веб-файрволів, систем виявлення вторгнень (IDS) та включення безпечних програмних патчів.
- Географічне розподілення навантаження: розміщення серверів у різних географічних областях та розподіл навантаження між ними може знизити вплив DDoS-атаки на один центральний сервер. Розподілення навантаження дозволяє розподілити трафік між кількома серверами і збільшити загальну доступність.
- Системи виявлення DDoS-атак: використання спеціалізованих систем виявлення DDoS-атак дозволяє реагувати на аномальний трафік і надавати інформацію для подальшої аналізу та блокування шкідливого трафіку.

Використання комбінації цих методів та технологій може допомогти підвищити рівень захисту на рівні застосунків від DDoS-атак та забезпечити нормальну роботу системи навіть під час атаки.

1.9 Географічне розподілення

Географічне розподілення серверів та інфраструктури є ефективним методом захисту від DDoS-атак. Цей підхід полягає в розміщенні серверів у різних географічних регіонах або у вузлах мережі, що знаходяться віддалено один від одного. Це дозволяє розподілити навантаження та забезпечити нормальну роботу навіть під час DDoS-атаки. Основний принцип роботи географічного розподілення включає наступні кроки:

1. Розташування серверів: сервери розміщуються у різних географічних регіонах, віддалених один від одного. Це можуть бути дата-центри, розташовані в різних країнах або регіонах.
2. Глобальна мережа доставки контенту (CDN): використання CDN дозволяє розподілити контент та обробку запитів між серверами, розташованими у

різних регіонах. CDN автоматично направляє запити користувачів до найближчого сервера, що знижує затримки та покращує швидкість доступу.

3. DNS-розподілення: використання глобально розподілених DNS-серверів дозволяє направляти запити до найближчих серверів з врахуванням географічного розташування користувача. Це допомагає зменшити затримки та покращити доступність.

4. Моніторинг та маршрутизація: мережеві пристрої, такі як маршрутизатори, моніторять навантаження на сервери та налаштовують маршрутизацію трафіку. При DDoS-атаках, коли один сервер перевантажений, маршрутизатори можуть перенаправляти трафік до інших доступних серверів, забезпечуючи нормальну роботу системи.

5. Системи виявлення DDoS-атак: встановлення систем виявлення DDoS-атак дозволяє швидко виявляти та реагувати на атаки. Це можуть бути спеціалізовані рішення, які аналізують трафік, виявляють аномалії та вживають заходів для мінімізації впливу атаки.

Цей метод захисту від DDoS-атак є досить популярним в наш час. Нижче наведено деякі переваги географічного розподілення:

- Розподілення навантаження: розташування серверів у різних регіонах дозволяє розподілити навантаження між ними. Під час DDoS-атаки, коли один сервер стає метою атаки і перевантажується, інші сервери можуть продовжувати обслуговувати запити, забезпечуючи нормальну доступність.
- Географічна віддаленість: розташування серверів в різних географічних регіонах зменшує вплив DDoS-атаки на окремі ресурси. Атакуючому складніше вплинути на всі сервери одночасно, оскільки вони знаходяться у різних фізичних місцях.
- Трафіковий розподіл: Використання Content Delivery Network (CDN) або Load Balancer дозволяє розподілити трафік між різними серверами в різних регіонах. Це дозволяє знизити навантаження на окремі сервери та забезпечити кращу доступність.

- Запобігання локальним атакам: якщо DDoS-атака спрямована на конкретний регіон або мережу, розподілення серверів може допомогти запобігти впливу атаки на всю інфраструктуру. Атакуючий не зможе сконцентрувати увагу на одному місці і отримати повний контроль.
- Більша стійкість до витоку даних: географічне розподілення серверів може знизити ризик витоку даних у разі успішної атаки. Якщо один сервер стає компромісованим, викрадення всієї інформації стає складнішим завданням через розподілену природу інфраструктури.

Загальна ідея географічного розподілення полягає в тому, щоб розподілити ресурси та трафік між різними серверами, що знаходяться у різних географічних регіонах. Це дозволяє покращити доступність, зменшити вплив DDoS-атак та забезпечити стійкість роботи системи навіть у разі атаки.

Важливо враховувати, що географічне розподілення само по собі не гарантує повністю захищену від DDoS-атак інфраструктуру, але це один з ефективних методів, який можна використовувати в комбінації з іншими заходами захисту для створення більш стійкої системи.

1.10 Хмарні рішення

Хмарні рішення для захисту від DDoS-атак - це метод захисту, при якому використовується хмарна інфраструктура для фільтрації та обробки трафіку, спрямованого на цільовий об'єкт атаки. Замість того, щоб пропускати весь трафік через внутрішні мережеві ресурси, хмарні рішення використовують розподілені хмарні сервери, які здатні швидко виявляти та фільтрувати вразливий трафік.

Основні риси хмарних рішень для захисту від DDoS-атак:

- Інфраструктура великого масштабу: хмарні провайдери мають розгалужені мережі серверів у різних географічних регіонах, що дозволяє розподілити навантаження та витримувати інтенсивні DDoS-атаки.

- Скорочення локального трафіку: хмарні рішення дозволяють обробляти трафік ближче до його джерела, мінімізуючи вплив атаки на внутрішню мережу та сервери.
- Обробка в реальному часі: хмарні рішення використовують розподілені системи виявлення та фільтрації для швидкого виявлення та блокування DDoS-атак у режимі реального часу.
- Еластичність та масштабованість: хмарні рішення можуть легко масштабуватись, щоб впоратись зі збільшеним трафіком під час атаки. Ресурси можуть бути автоматично розширені або зменшені, забезпечуючи потрібну ємність для захисту.
- Гнучкість та настроювання: хмарні рішення дозволяють налаштовувати правила фільтрації та налаштування для кожного клієнта окремо, враховуючи їхні специфічні потреби та вимоги.
- Системи аналізу та моніторингу: хмарні рішення надають засоби для аналізу та моніторингу трафіку, що дозволяє виявляти аномалії та спостерігати за виявленням потенційних DDoS-атак.

Даний тип захисту пропонує широкий спектр функцій та можливостей для виявлення, моніторингу та фільтрації вразливого трафіку, що надходить на цільовий об'єкт атаки. Основні переваги хмарних рішень для захисту від DDoS-атак включають:

- Масштабованість: хмарні рішення здатні швидко масштабуватись та обробляти великі обсяги трафіку, що дозволяє ефективно захищати від DDoS-атак навіть у разі інтенсивних навантажень.
- Гнучкість: хмарні рішення надають можливість налаштування та кастомізації захисту відповідно до потреб і вимог конкретної організації. Це включає налаштування правил фільтрації, контролю доступу та інших параметрів.
- Глобальний захист: хмарні рішення мають географічно розподілені сервери та точки присутності по всьому світу, що дозволяє захиститись від DDoS-

атак у будь-якій локації. Це дозволяє розподіляти навантаження та забезпечувати доступність сервісу навіть під час атаки.

- Аналіз трафіку: хмарні рішення використовують розширені аналітичні алгоритми для виявлення аномалій у трафіку та ідентифікації DDoS-атак. Вони можуть швидко реагувати та автоматично блокувати шкідливий трафік.
- Системи інтелектуального розпізнавання: хмарні рішення використовують системи інтелектуального розпізнавання для відрізнення нормального трафіку від шкідливого. Вони використовують машинне навчання та аналіз статистичних даних для виявлення та блокування DDoS-атак.
- Керування трафіком: хмарні рішення можуть управляти трафіком та розподіляти його між різними ресурсами, що дозволяє розподілити навантаження та забезпечити стійкість роботи системи під час атаки.

Хмарні рішення можуть бути дуже ефективними у захисті від DDoS-атак, оскільки вони надають широкий спектр засобів захисту та можливості масштабування. Хмарні провайдери зазвичай мають команди експертів з безпеки, які постійно моніторять та аналізують трафік, щоб виявляти атаки та розробляти стратегії захисту. Вони можуть швидко реагувати на нові типи атак і надавати вам підтримку в управлінні DDoS-захистом.

Використання хмарних рішень для захисту від DDoS-атак дозволяє ефективно захищати свою мережу та інфраструктуру, забезпечуючи стійкість роботи та доступність сервісів навіть під час інтенсивних атак. Хмарні рішення можуть працювати в поєднанні з іншими методами захисту від DDoS-атак, такими як мережеві екрани, аналізатори пакетів, системи моніторингу тощо. Кожне рішення має свої унікальні особливості та переваги, і вибір підходящого хмарного рішення залежить від конкретних потреб та вимог вашої організації.

Варто зазначити, що хмарні рішення також мають свої обмеження і вимагають правильного конфігурування та управління для ефективного захисту. Рекомендується проконсультуватись з провайдером хмарних послуг та спеціалістами з безпеки для визначення найкращих практик і стратегій захисту в конкретному випадку.

Порівняння хмарних рішень наведено в таблиці 1.1.

Таблиця 1.1

Порівняння хмарних сервісів

Cloudflare	Akamai Technologies	Imperva Incapsula	F5
Забезпечує послуги CDN та захисту від DDoS-атак.	Забезпечує послуги CDN та захисту від DDoS-атак.	Забезпечує послуги CDN та захисту від DDoS-атак.	Забезпечує послуги CDN та захисту від DDoS-атак.
Широка глобальна мережа серверів для розподілення трафіку та зменшення впливу атаки.	Розподілені сервери для зменшення впливу атаки та забезпечення швидкості доставки контенту.	Ефективне виявлення та блокування DDoS-атак на рівні мережі та застосунків.	Має багаторівневий захист, який здатний відбивати DDoS атаки на мережевому, сеансовому та прикладному рівнях.
Велика кількість додаткових функцій, таких як SSL-шифрування, брандмауери, захист від ботів тощо	Висока масштабованість та надійність.	Велика мережа серверів та висока швидкість доставки контенту.	Може масштабуватися, щоб відповідати зростаючому навантаженню або атакам DDoS великого масштабу.
Простий у використанні інтерфейс та широкий спектр планів ціноутворення	Додаткові функції, такі як брандмауери, контроль доступу, захист від вразливостей.	Додаткові функції, такі як захист від ботів, кешування та аналіз трафіку.	Детальний контроль трафіку з функціями, такими як глибокий аналіз пакетів, огляд SSL та шифрування.

Підводячи підсумок, можна стверджувати, що організації можуть значно знизити бізнес-ризики та зменшити ймовірність простою, обираючи хмарні служби захисту від DDoS-атак. Хмарний захист від атак також дозволяє організаціям проактивно захищатися від нових і невідомих загроз, які постійно оновлюються постачальниками керованих послуг.

1.11 Балансування навантаження

Балансування навантаження (Load Balancing) є одним із методів захисту від DDoS-атак, що дозволяє розподілити навантаження між декількома серверами або ресурсами для забезпечення високої доступності та стійкості системи. Балансування навантаження може бути ефективним інструментом протидії DDoS-атакам, оскільки дозволяє розподілити навантаження на кілька ресурсів, надаючи більшу пропускну здатність та витривалість до атак.

Основна ідея балансування навантаження полягає в тому, що навантаження розподіляється між кількома серверами або ресурсами за допомогою спеціального пристрою (балансувальника навантаження) або програмного забезпечення. Цей пристрій аналізує трафік та розподіляє його між доступними ресурсами залежно від певних правил або алгоритмів.

В контексті захисту від DDoS-атак, балансування навантаження може мати наступні переваги:

- Розподілення навантаження: балансування навантаження дозволяє розподілити трафік між кількома серверами або ресурсами. Це зменшує ризик перевантаження одного конкретного ресурсу під час DDoS-атаки і дозволяє системі продовжувати працювати і обслуговувати легітимний трафік.
- Захист від перевантаження: балансування навантаження розподіляє трафік між різними ресурсами, що забезпечує більшу пропускну здатність та витривалість системи під час DDoS-атаки. В разі, якщо один ресурс стає

перевантаженим атакою, інші ресурси можуть продовжувати працювати і обробляти трафік.

- Фільтрація шкідливого трафіку: балансувальники навантаження можуть використовувати різні алгоритми для фільтрації шкідливого трафіку, який генерується DDoS-атакою. Наприклад, на основі IP-адрес атакуючих агентів або аналізу характеристик пакетів. Це допомагає відсікти шкідливий трафік і захистити ресурси від атак.

Загалом, балансування навантаження є важливим компонентом захисту від DDoS-атак, оскільки допомагає розподілити навантаження та захистити ресурси від перевантаження та відмови в обслуговуванні під час атаки.

1.12 Моніторинг та виявлення

Моніторинг та виявлення є важливою складовою захисту від DDoS-атак і допомагають вчасно виявляти та реагувати на атаки. Основна мета моніторингу - це постійне спостереження за мережею та трафіком для виявлення аномальних показників, які можуть вказувати на наявність DDoS-атаки. Деякі переваги моніторингу та виявлення включають:

- Виявлення аномалій: системи моніторингу використовують різні метрики та алгоритми для аналізу трафіку та виявлення аномальних змін, таких як зростання обсягу трафіку, незвичайний розподіл джерелової адреси, патернів атак та інших несподіваних змін.
- Швидка реакція: правильно налаштовані системи моніторингу можуть виявити атаку на ранніх стадіях і сповістити адміністраторів про потенційну загрозу. Це дозволяє здійснити швидку реакцію та прийняти відповідні заходи для забезпечення безпеки системи.
- Візуалізація даних: багато систем моніторингу надають інтерфейси та звіти, які допомагають візуалізувати дані про мережу та трафік. Це дозволяє

адміністраторам швидко аналізувати та розуміти поточний стан мережі, виявляти аномалії та реагувати на них.

- Інтеграція з іншими системами захисту: системи моніторингу часто можуть інтегруватися з іншими інструментами та системами захисту, такими як файрволи, системи виявлення вторгнень (IDS), системи балансування навантаження та інші. Це дозволяє автоматично запускати захисні механізми та реагувати на виявлені загрози.

Загалом, моніторинг та виявлення грають важливу роль в захисті від DDoS-атак, допомагаючи вчасно розпізнавати аномалії та інциденти безпеки для швидкого реагування і зменшення можливих наслідків атаки.

1.13 Масштабованість і резервне копіювання

Важливими аспектами захисту від DDoS-атак є масштабованість і резервне копіювання. Використання масштабованості та резервного копіювання дозволяє забезпечити надійний захист від DDoS-атак, зменшити вплив атак на доступність та стабільність системи і забезпечити швидке відновлення після атаки.

Масштабованість означає, що інфраструктура та ресурси можуть розширюватися або зменшуватися залежно від обсягу трафіку або навантаження. У контексті захисту від DDoS-атак, масштабованість дозволяє розподілити навантаження між багатьма серверами або ресурсами, що дозволяє краще витримувати атаку та забезпечувати доступ до послуг для легітимних користувачів. Вона може бути вертикальною (додавання ресурсів до одного вузла, наприклад, більше ОЗУ або більш потужний процесор) або горизонтальною (додавання нових вузлів до системи).

Резервне копіювання даних і налаштувань є важливим елементом захисту від DDoS-атак. Це процес копіювання даних для їх зберігання в іншому місці. Регулярне створення резервних копій дозволяє відновити систему після атаки та запобігти втраті даних або відновити працездатність системи в якомусь резервному середовищі.

Резервне копіювання також дозволяє відновити налаштування та політики безпеки після атаки.

Масштабованість і резервне копіювання дозволяють розподілити ризик та забезпечити витримку системи під час DDoS-атаки. Розподіл ресурсів та даних між кількома серверами або регіонами дозволяє зменшити вплив атаки на систему в цілому і забезпечити більшу стійкість до атак.

Висновки за розділом 1

Розподілені атаки типу «відмова в обслуговуванні» сьогодні викликають серйозне занепокоєння в Інтернеті. Напади стають все більш частими і складними. Компанії будь-якого розміру, фінансові установи, організації, уряди тощо є потенційними цілями DDoS-атак. Вкрай необхідно застосувати методи боротьби з DDoS і боротися з іншими проблемами кібербезпеки.

DDoS-атаки спрямовані на перевантаження ресурсів системи, що призводить до відмови в обслуговуванні для законних користувачів. Це може призвести до серйозного зниження доступності вебресурсу, що впливає на довіру користувачів, репутацію бренду та потенційні втрати доходу. Боротьба з DDoS-атаками допомагає забезпечити нормальну доступність і функціональність системи. У деяких випадках DDoS-атаки можуть бути спрямовані на особливо конкурентні бренди або на організації з метою завдання шкоди або спроби використати конкурентну перевагу.

Загальний висновок полягає в тому, що захист від DDoS-атак вимагає комплексного підходу та використання різних методів на різних рівнях системи. Комбінація захисних заходів на рівні мережі, рівні застосунків, використання хмарних рішень та аналізу загроз може допомогти зменшити ризики та ефективно захищати систему від DDoS-атак.

РОЗДІЛ 2
РЕКОМЕНДАЦІЇ ЩОДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз загроз

Першу рекомендацію щодо розробки програмного забезпечення необхідно розпочати з аналізу загроз та існуючих типів DDoS-атак. Розуміння цих загроз допоможе нам визначити потрібні функціональні можливості майбутнього програмного модуля для захисту від атак на відмову в обслуговуванні.

DDoS-атаки створюють різні загрози для організацій та інфраструктури. Деякі загрози, пов'язані з DDoS-атаками, включають:

- Втрата доступності: головна загроза полягає в тому, що DDoS-атаки спрямовані на перевантаження ресурсів системи, що може призвести до тимчасового або повного відмови у доступі до сервісів для законних користувачів.
- Втрата продуктивності: DDoS-атаки можуть затримувати роботу системи, спричиняти погіршення її продуктивності та знижувати швидкість реакції.
- Витрати на пропускну здатність: атаки можуть призвести до значного збільшення мережевого трафіку, що вимагатиме додаткових інвестицій у пропускну здатність мережі та обчислювальну інфраструктуру.
- Виток конфіденційної інформації: деякі DDoS-атаки можуть бути використані як відволікаючий маневр для отримання несанкціонованого доступу до системи та викрадення конфіденційних даних.
- Втрата доходу та репутації: довготривала або повторювана DDoS-атака може призвести до втрати доходу і клієнтів, а також позначитися на репутації організації.
- Збої в роботі сервісів: атаки можуть призвести до неполадок в роботі веб-сайтів, онлайн-платформ, додатків та інших сервісів.

- Злочинні дії: DDoS-атаки можуть бути використані як засіб вторгнення або відволікання уваги від інших кіберзлочинних дій, таких як викрадення даних або злам системи.

Ці загрози показують, наскільки серйозними можуть бути DDoS-атаки і чому важливо приділяти належну увагу захисту від них. Аналіз загроз є важливим етапом в розробці ефективних заходів захисту. Він передбачає враховування таких аспектів:

- Типи DDoS-атак: необхідно вивчити різні типи DDoS-атак, такі як SYN флуд, UDP флуд, HTTP флуд, ICMP флуд тощо. Кожен тип атаки має свої особливості і вплив на систему, тому важливо зрозуміти їхні механізми дії.
- Об'єкти атак: визначення ресурсів та компонентів системи, які можуть стати об'єктами атаки. Це можуть бути мережеві сервіси, сервери, маршрутизатори, додатки або інші компоненти інфраструктури.
- Рівні атак: розгляд різних рівнів атак, такі як мережевий рівень, транспортний рівень або рівень застосунків. Кожен рівень може бути підданому специфічним видам атак і вимагати окремих захисних стратегій.
- Масштаб атак: визначення потенційного масштабу атаки, включаючи обсяг трафіку, кількість атакуючих джерел, тривалість атаки тощо. Це допоможе оцінити потенційний вплив атаки на інфраструктуру системи.
- Вектори атак: дослідження різних векторів атак, які можуть бути використані для запуску DDoS-атаки. Це можуть бути напрямлені атаки на певні слабкі місця системи, використання ампліфікації або використання ботнетів.
- Способи виявлення: розгляд методів виявлення DDoS-атак, включаючи аналіз мережевого трафіку, моніторинг системних параметрів, аналіз журналів подій та використання алгоритмів машинного навчання.
- Вплив на бізнес: передбачення можливих наслідків DDoS-атак для бізнесу, такі як втрата доступності, втрати фінансових коштів, пошкодження репутації тощо. Це допоможе визначити рівень прийнятного ризику та обґрунтувати необхідність заходів захисту.

2.2 Фільтрація трафіку

Фільтрація трафіку - це процес контролю доступу до мережі, який використовується для захисту мережі від небажаного або шкідливого трафіку. Це може бути здійснено шляхом аналізу даних, що проходять через мережу, і відсівання трафіку, який не відповідає встановленим критеріям безпеки. Даний процес є важливим компонентом управління мережевим трафіком і мережевої безпеки.

Фільтрація трафіку може бути здійснена за допомогою різних критеріїв, таких як:

- IP-адреса: блокування трафіку з конкретних IP-адрес або діапазонів IP-адрес.
- Порт: блокування трафіку на конкретних портах. Наприклад, можна заблокувати всі вхідні з'єднання на порт 80, щоб запобігти небажаним веб-запитам.
- Протокол: Блокування трафіку відповідно до протоколу (наприклад, TCP, UDP, ICMP і т.д.).
- Вміст пакета: фільтрація трафіку може бути здійснена за допомогою глибокого аналізу пакетів (Deep Packet Inspection, DPI), який вимагає аналізу вмісту пакетів, а не просто заголовків пакетів.
- Час: можна фільтрувати трафік в залежності від часу дня або дня тижня.

Фільтрація трафіку допомагає забезпечити безпеку мережі, запобігаючи небажаному або шкідливому трафіку, такому як спроби вторгнення, DDoS-атаки або спам. Крім того, вона може допомогти оптимізувати пропускну здатність мережі, обмежуючи трафік, який не є критичним для бізнесу.

2.3 Зм'якшення атак

Зм'якшення атак – це процес зменшення або усунення негативних ефектів або ризиків. Термін часто використовується в контексті управління ризиками,

катастрофами, зміною клімату, технологіями інформаційної безпеки та іншими областями.

В контексті технологій інформаційної безпеки, зм'якшення часто відноситься до заходів, призначених для зменшення впливу або ймовірності шкідливих подій, таких як кібератаки або витоки даних.

Зм'якшення атак – це стратегія, яка спрямована на зменшення впливу або зниження наслідків атаки. Вона може бути важливою частиною плану захисту інформаційної безпеки будь-якої організації. Зм'якшення атак зазвичай включає такі дії:

- Виявлення: швидке виявлення атаки є критично важливим для зменшення її впливу. Це може включати моніторинг мережевого трафіку, систем аудиту, застосування системи виявлення вторгнень (IDS) і системи запобігання вторгненням (IPS).
- Аналіз: Після виявлення атаки, слід провести аналіз, щоб визначити джерело атаки, її мету, а також методи, що були використані зловмисником. Це допоможе розробити стратегію зм'якшення.
- Реагування: це може включати в себе такі дії, як відключення атакованих систем, ізоляція вражених мереж, припинення шкідливого трафіку або застосування патчів для виправлення вразливостей.
- Відновлення: після того, як атака була зупинена, необхідно відновити сервіси і дані. Це може включати в себе відновлення даних з резервних копій, відновлення систем до безпечного стану або заміну компрометованого обладнання.
- Навчання: після атаки важливо зрозуміти, що сталося, і внести зміни в політику безпеки або процедури, щоб запобігти подібним атакам в майбутньому.

Зм'якшення атак повинна бути частиною загальної стратегії безпеки, що також включає запобігання атакам і підготовку до можливих інцидентів безпеки.

В контексті захисту від DDoS-атак зм'якшення включає в себе ряд заходів для зменшення або усунення наслідків таких атак. Це може включати в себе використання спеціалізованих систем, які розпізнають та блокують DDoS-атаки, а також більш

загальні підходи до управління мережевим трафіком. До стратегій стратегій зм'якшення DDoS-атак можна віднести:

- Управління пропускною здатністю: Одним з найпростіших способів боротьби з DDoS-атаками є забезпечення достатньої пропускної здатності, щоб впоратися з піковими навантаженнями. Це не завжди практично, особливо відносно великих атак, але це може допомогти уникнути проблем з менш важкими атаками.
- Обмеження швидкості: дана операція може зменшити вплив DDoS-атак, обмежуючи кількість запитів, які можна зробити за певний період часу.
- Захист на рівні аплікації: деякі DDoS-атаки цілять в слабкі місця на рівні програмного забезпечення. Заходи безпеки на рівні програми можуть включати в себе використання останніх патчей безпеки, використання Web Application Firewalls (WAF), або зміну конфігурації сервера для обмеження вразливості до атак.
- Використання спеціалізованих послуг DDoS-зм'якшення: існує багато компаній, що надають послуги для захисту від DDoS-атак. Ці послуги, зазвичай, включають в себе набір технологій, що допомагають виявляти та блокувати DDoS-атаки на ранніх стадіях.

Зм'якшенняDDoS-атак - це постійна боротьба, оскільки зловмисники постійно розвивають нові стратегії атак. Однак, за допомогою правильних стратегій і технологій, можна значно зменшити ризик і вплив таких атак.

2.4 Мережевий екран

Мережевий екран, або firewall, це система безпеки, що використовується для контролю вхідного і вихідного мережевого трафіку на основі визначених правил безпеки. Мережеві екрани створені для запобігання небажаного доступу до або з приватних мереж.

Мережеві екрани можуть бути апаратними або програмними, або ж комбінацією обох. Вони діють як бар'єр між захищеною і контрольованою внутрішньою мережею та менш надійною зовнішньою мережею, такою як Інтернет.

Вони аналізують пакети даних, що передаються через мережу, і вирішують, чи дозволити їм проходити, або заблокувати їх.

Ось декілька основних типів мережевих екранів:

- Packet Filtering Firewalls: найпростіший тип мережевого екрана. Вони перевіряють пакети даних, що приходять і виходять з мережі, і або блокують, або дозволяють їх на основі встановлених правил безпеки.
- Stateful Inspection Firewalls: вони перевіряють не тільки пакети даних, але і стан з'єднання, в якому вони приходять або виходять. Це дозволяє їм робити більш глибокі висновки про трафік.
- Proxy Firewalls: цей тип екрана робить як посередника між користувачем та службою, до якої вони намагаються отримати доступ, що дозволяє краще контролювати і фільтрувати трафік.
- Next-Generation Firewalls (NGFWs): NGFWs комбінують функції традиційних мережевих екранів з іншими системами оборони, такими як системи виявлення вторгнень (IDS) і системи запобігання вторгненням (IPS).

Важливо зауважити, що мережеві екрани використовуються не тільки для блокування вхідного трафіку, але і вихідного. Це може допомогти забезпечити, що шкідливі або небажані програми всередині мережі не можуть надсилати дані назовні.

Програмний модуль забезпечення захисту вебзастосунків від DDoS-атак буде функціонувати на ОС Linux, тому варто переглянути мережеві екрани, які може підтримувати дана система.

Linux має вбудований мережевий екран у ядро системи, відомий як Netfilter, що функціонує за допомогою інструменту командного рядка iptables або новішого nftables.

- Iptables – це засіб командного рядка, що використовується для налаштування правил Linux Netfilter. Iptables надзвичайно потужний і гнучкий, але може бути складним для нових користувачів.
- Nftables – це новіший інструмент, розроблений як наступник Iptables, надаючи більше гнучкості і зрозумілості.

Для більш простого інтерфейсу, є кілька інших рішень, які надають графічний інтерфейс користувача або спрощують синтаксис Iptables/Nftables:

1. UFW (Uncomplicated Firewall): UFW розроблений з метою полегшення керування firewall в Linux. Він використовує Iptables під капотом, але надає більш простий для користувача інтерфейс.

2. Firewalld: Firewalld є стандартним фаєрволом для декількох дистрибутивів Linux, включаючи Fedora і RHEL/CentOS. Він надає динамічне управління правилами без необхідності перезавантажувати всю політику екрана.

3. Shorewall: Shorewall, або "Shoreline Firewall", є високорівневим інструментом для налаштування фаєрволу на Linux. Він розроблений для облегшення керування складними конфігураціями фаєрволу.

4. Gufw: Gufw є графічним інтерфейсом для UFW і є добрим вибором для користувачів, які віддають перевагу графічному інтерфейсу.

2.5 Чорний список IP-адрес

Чорний список IP-адрес - це список IP-адрес, які були ідентифіковані як ненадійні або зловмисні. Цей список може бути використаний для блокування спроб з'єднання від цих IP-адрес, що захищає систему або мережу від потенційно шкідливого трафіку. Блокування за IP-адресою може бути особливо ефективним проти DDoS-атак або інших вторгнень в мережу.

Однак важливо пам'ятати, що IP-адреси можуть бути змінені або маскуватися, тому цей метод не є панацеєю від всіх видів атак. Мережеві екрани, такі як iptables в Linux, дозволяють створювати чорні списки IP-адрес і автоматично блокувати вхідний трафік від цих адрес.

Якщо IP-адреса знаходиться у чорному списку, вона зазвичай не матиме змоги встановлювати з'єднання з мережею або системою, в залежності від того, як ми налаштували свій мережевий екран.

Використання чорного списку полягає у наступному:

1. Ідентифікація атакуючих IP-адрес: перший крок полягає у визначенні IP-адрес, з яких надходять DDoS-атаки. Це можна зробити за допомогою моніторингу трафіку, аналізу логів сервера або використання спеціалізованих інструментів для виявлення DDoS-атак.

2. Додавання IP-адрес до чорного списку: після визначення атакуючих IP-адрес, їх можна додати до чорного списку в мережевому екрані або системі захисту від DDoS. Зазвичай це включає в себе налаштування правил, що блокують вхідний трафік від цих IP-адрес.

3. Автоматичне блокування IP-адрес: деякі системи захисту від DDoS мають можливість автоматичного додавання IP-адрес до чорного списку на основі поведінки трафіку. Наприклад, якщо від певної адреси надходить аномально великий обсяг трафіку, система може автоматично додати цю адресу до чорного списку.

Важливо зазначити, що використання чорного списку IP-адрес може бути лише одним з компонентів комплексного плану захисту від DDoS. Оскільки атакуючі можуть використовувати змінні IP-адреси або ботнети з тисячами комп'ютерів, може бути неможливо заблокувати всі атакуючі IP-адреси.

Тому важливо також розглядати інші стратегії захисту, такі як збільшення пропускної здатності, використання CDN або використання спеціалізованих сервісів захисту від DDoS.

2.6 Захист на рівні програмного забезпечення

Захист на рівні програмного забезпечення , також відомий як захист від атак на сьомому рівні моделі OSI, зосереджений на специфічних веб-додатках або службах, які цільові атаки намагаються виключити. DDoS-атаки на прикладному рівні часто використовують дуже вибіркові і розраховані тактики, щоб сповільнити або зупинити роботу конкретного додатка. Вони можуть включати в себе масові запити до конкретної веб-сторінки, бази даних або API, з метою перевантаження цих служб і виклику перебоїв у роботі.

Використовуючи PHP, ми можемо розробити просту систему для відслідковування та обмеження кількості запитів до вашого сервера. PHP як мова програмування застосовується в основному на рівні додатку, тому для захисту від DDoS-атак можна реалізувати декілька різних стратегій. Така система може допомогти нам ідентифікувати потенційні DDoS-атаки. Ось один з можливих підходів:

- Відслідковування IP-адрес: за допомогою глобальної змінної $_SERVER, ми можемо отримати IP-адресу, яка надіслала запит.
- Відслідковування кількості запитів: ми можемо створити систему, яка відслідковує кількість запитів від кожної IP-адреси за певний проміжок часу. Це може бути реалізовано з використанням бази даних або файлу, куди ми записуємо IP-адресу та час кожного запиту.
- Обмеження кількості запитів: якщо кількість запитів від певної IP-адреси перевищує певний поріг за певний проміжок часу, ми можемо відхилити додаткові запити від цієї IP-адреси.

На рисунку 1.2 наведений простий приклад коду PHP для цього:

```php
<?php
session_start();

if (!isset($_SESSION['LAST_CALL'])) {
    $_SESSION['LAST_CALL'] = time();
} else if (time() - $_SESSION['LAST_CALL'] < 3) {
    http_response_code(429);
    exit("Ви надто часто відвідуєте цю сторінку. Будь ласка, зачекайте кілька секунд.");
}

$_SESSION['LAST_CALL'] = time();

// Продовження коду
```

Рисунок 1.2 – код PHP для захисту від DDoS-атак

У цьому прикладі коду ми використовуємо сесії для відслідковування часу останнього запиту від користувача. Якщо користувач робить запити занадто часто (у цьому випадку більше ніж один раз кожні 3 секунди), ми відхиляємо запит.

Важливо пам'ятати, що цей метод є простим і не надасть повного захисту від усіх типів DDoS-атак. Це може бути лише частиною більш комплексного плану захисту. Мова програмування PHP - це лише один інструмент у вашому арсеналі. Ви також повинні розглянути можливість використання мережевих заходів безпеки, таких як файрволи та системи виявлення вторгнень, а також заходи на рівні веб-сервера, наприклад, модулі, які забезпечують захист від DDoS-атак.

Висновки за розділом 2

Загалом, розробка програмного забезпечення для захисту від DDoS-атак вимагає комплексного підходу, аналізу загроз, використання різноманітних методів захисту та постійного оновлення. Забезпечення доступності системи та захисту від DDoS-атак є важливими аспектами для забезпечення безпеки, стійкості та успішної роботи системи.

Ретельний аналіз типів та методів DDoS-атак, що можуть спрямовуватись на систему, є важливим кроком у розробці програмного забезпечення для захисту. Це допомагає розуміти потенційні загрози і визначати ефективні стратегії захисту.

При розробці рішення для захисту необхідно використовувати різноманітні заходи, такі як фільтрація трафіку, обмеження швидкості, моніторинг та виявлення аномалій, резервне копіювання та відновлення, використання хмарних рішень тощо. Комбінація цих заходів дозволяє ефективно відповідати на DDoS-атаки.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ЗАХИСТУ ВІД DDOS-АТАК

3.1 Алгоритм ідентифікації DDoS-атаки

Ідентифікація DDoS-атаки може бути складним процесом, тому що ці атаки часто маскуються під легітимний трафік. Однак, існують декілька загальних показників, які можуть вказувати на потенційну DDoS-атаку:

- Різке збільшення трафіку: якщо було помічено раптове збільшення вхідного трафіку або числа запитів до певного ресурсу, це може бути показником DDoS-атаки.
- Постійні запити з одного або кількох джерел: якщо було помічено, що велика кількість запитів приходить з одного або кількох IP-адрес, це може бути ознакою DDoS-атаки.
- Неякісна робота додатка або повільність: якщо було помічено, що вебсайт або додаток стає повільним або відповідає на запити з помилками, це також може бути показником DDoS-атаки.

Алгоритм ідентифікації DDoS-атаки може включати наступні кроки:

- Моніторинг трафіку: ви повинні постійно спостерігати за своїм трафіком, щоб виявити несподівані зміни.
- Аналіз джерел трафіку: дивіться на джерела трафіку та типи запитів, які ви отримуєте. Чи є вони легітимними?
- Визначення аномалій: використовуйте інструменти аналітики та виявлення вторгнень для визначення аномалій в трафіку або поведінці користувачів.
- Реагування на підозрілу активність: якщо ви виявите підозрілу активність, вживайте заходів, щоб заблокувати або обмежити її.

Це лише загальний підхід, який допоможе налаштувати свої стратегії відповідно до конкретного середовища та потреб.

Враховуючи наведені вище рекомендації, можна створити блок-схему алгоритму ідентифікації DDoS-атаки. Вона наведена на рисунку 1.3.

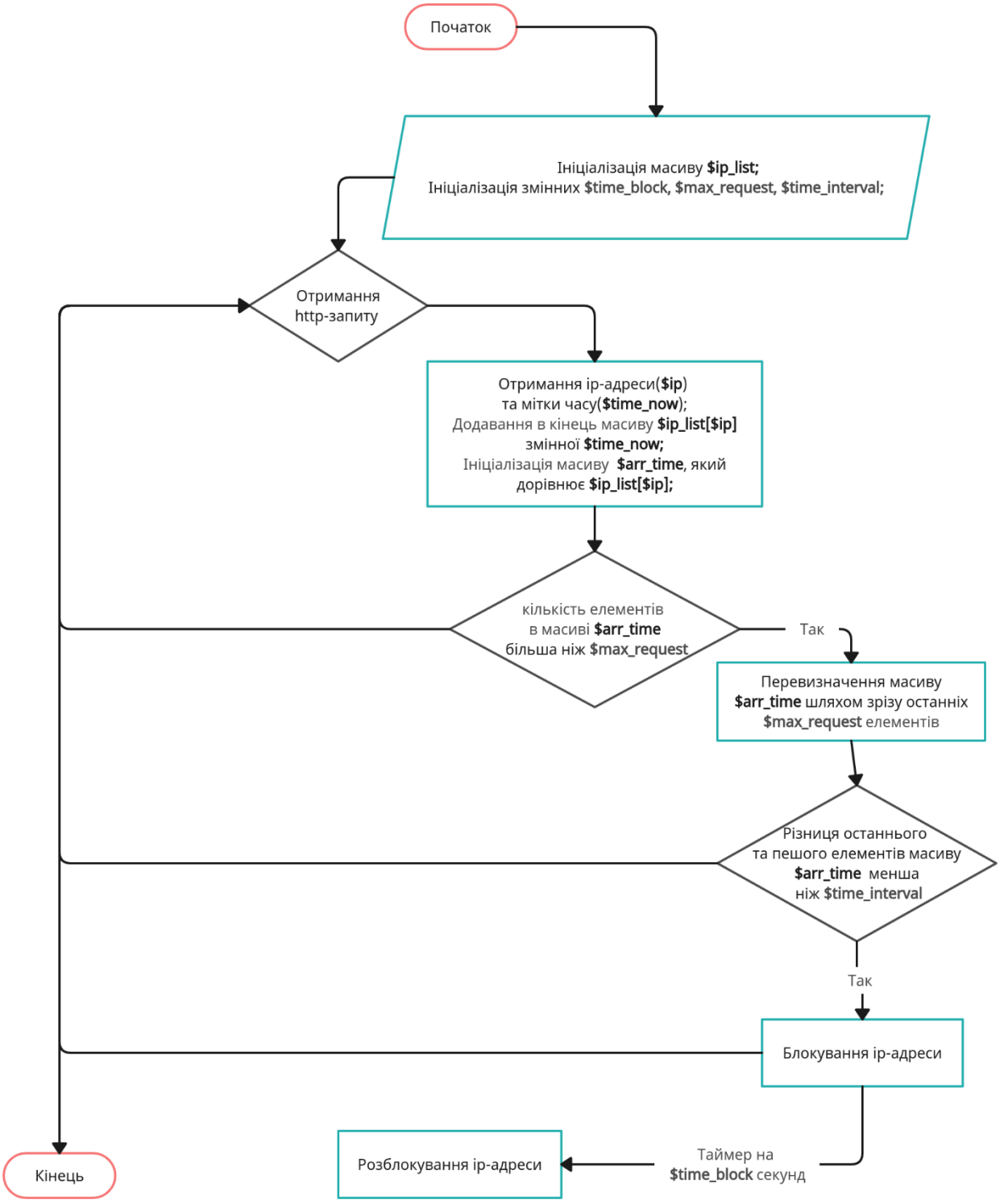

Рисунок 1.3 – блок-схема алгоритму ідентифікації DDoS-атаки

Її можна розділити на наступні кроки:

1. Визначення конфігураційних змінних програмного модуля.
2. Отримання IP-адреси клієнта та поточного часу. Додавання даних параметрів у сховище(масив).
3. Перевірка даної IP-адреси на перевищення кількості запитів.
4. Перевірка кількості запитів за визначений проміжок часу.
5. Блокування даної IP-адреси та створення відкладеної задачі на її розблокування.

3.2 Механізм реагування на атаку

При виявленні DDoS-атаки необхідно швидко реагувати, щоб мінімізувати потенційні збитки. Мережевий екран може бути корисним інструментом для реагування на атаки. У Linux існує кілька мережевих екранів, які можна використовувати для управління мережевим трафіком та забезпечення безпеки. Ось декілька з них:

- Iptables – це традиційний мережевий екран для Linux. Iptables дозволяє адміністраторам встановлювати правила для вхідного та вихідного трафіку, що проходить через систему.
- Nftables – це проект, який призначений для заміни iptables. Nftables надає більш сучасний синтаксис та включає в себе додаткові можливості, такі як вбудований механізм нативних наборів даних, що полегшують створення складних правил.
- Firewalld – це мережевий екран, розроблений для дистрибутивів Linux, які використовують systemd, наприклад Fedora і Red Hat Enterprise Linux. Firewalld пропонує динамічне управління правилами без перезавантаження конфігурації та підтримує зони, які дозволяють розділити мережі на різні рівні довіри.

- UFW (Uncomplicated Firewall): UFW - це мережевий екран, розроблений для Ubuntu, щоб спростити процес управління мережевим екраном. Він надає простий синтаксис для управління iptables і підходить для нових користувачів Linux.
- IPFire – це Linux-дистрибутив, який спеціалізується на функціях мережевого екрана. Він включає в себе великий набір можливостей, включаючи проксі-сервери, VPN, IDS/IPS, QoS і багато іншого.

Для розробки програмного модуля забезпечення захисту вебзастосунків від DDoS-атак я вирішив використовувати мережевий екран iptables, опираючись на його простоту та багату документацію.

Iptables це система управління мережевими правилами в ядрах Linux, що базується на командному рядку. Вона використовується для налаштування, підтримки та керування мережевими маршрутами та правилами у операційній системі Linux. Дане рішення дозволяє адміністратору налаштовувати правила, що визначають, яким чином пакети (блоки даних) будуть оброблятися системою. Це включає в себе можливість дозволу або заборони трафіку в залежності від різних критеріїв, таких як IP-адреса джерела або призначення, тип пакета, порт, протокол і т.д. Також він може бути використаний для забезпечення функцій, таких як NAT (Network Address Translation) та PAT (Port Address Translation).

Хоча Iptables був стандартом для більшості дистрибутивів Linux протягом багатьох років, він почав змінюватися на більш нові інструменти, такі як Nftables або Firewalld, що пропонують покращений синтаксис і більш потужні можливості.

Ключові поняття Iptables:

- Таблиці: в Iptables існують чотири основні таблиці, які використовуються для різних цілей: таблиця filter (для фільтрації пакетів), таблиця nat (для перетворення мережевих адрес), таблиця mangle (для модифікації пакетів) та таблиця raw (для обробки пакетів, які необхідно проігнорувати стандартним механізмом відслідкування з'єднань).
- Ланцюжки: ланцюжки - це набори правил, які призначені для пакетів, що проходять через систему. Існує три стандартні ланцюжки: INPUT (для обробки

вхідного трафіку), OUTPUT (для обробки вихідного трафіку) та FORWARD (для обробки трафіку, що не призначений для локальної системи).

- Правила: правила в Iptables визначають, як будуть оброблятися пакети. Кожне правило містить критерій (або набір критеріїв) і дію, яку слід виконати, якщо пакет відповідає цим критеріям (наприклад, прийняти пакет, відкинути пакет або переслати пакет до іншого ланцюжка).
- Політики: кожен ланцюг Iptables має стандартну політику, яка визначає, яку дію слід виконати для пакетів, які не відповідають жодному з правил в ланцюгу.
- Модулі: Iptables має модулі, які додають додаткові функції або властивості. Наприклад, модуль state може використовуватися для відслідкування стану з'єднань, а модуль limit може використовуватися для обмеження кількості пакетів, що приймаються від певного джерела.
- Специфікація пакетів: Iptables дозволяє вказувати пакети на основі різних атрибутів, таких як IP-адреса джерела або призначення, порт, протокол (TCP, UDP, ICMP, тощо) і багато іншого.

Команда iptables використовує різні прапори (або опції), які дозволяють керувати її поведінкою. Ось декілька основних прапорів, які допоможуть нам вивести команди CLI для блокування та розблокування IP-адреси:

1. -A (append): додає нове правило в кінець вибраного ланцюжка.
2. -D (delete): видаляє одне або кілька правил з вибраного ланцюжка.
3. -I (insert): вставляє нове правило на певну позицію в ланцюжку.
4. -F (flush): видаляє всі правила з вибраного ланцюжка.
5. -L (list): показує всі правила в вибраному ланцюжку.
6. -N (new chain): створює новий користувацький ланцюжок.
7. -X (delete chain): видаляє вибраний користувацький ланцюжок.
8. -P (default policy): встановлює політику за замовчуванням для вибраного ланцюжка.
9. -s (source): вказує IP-адресу джерела для правила.
10. -d (destination): вказує IP-адресу призначення для правила.

11. -p (protocol): вказує протокол (наприклад, tcp, udp, icmp) для правила.

12. -j (jump): вказує дію, яку слід виконати, якщо пакет відповідає критеріям правила. Наприклад:

- ACCEPT – пропустити, передати на обробку наступному правилу;
- DROP – видалити, більше над пакетом ніякої обробки не проводиться;
- RETURN – повернути на аналіз у попередній ланцюжок.

Враховуючи наведені вище поняття та прапори, можна вивести дві команди для блокування та розблокування IP-адреси:

- iptables -A INPUT -s $ip -j DROP: буде використовуватись для блокування вхідного трафіку від джерела з IP-адресою $ip.
- iptables -D INPUT -s $ip -j DROP: буде використовуватись для розблокування вхідного трафіку від джерела з IP-адресою $ip. Дана команда видалить попередньо додане правило блокування через певний проміжок часу.

В мові програмування PHP присутня функція shell_exec - вона є вбудованою функцією, яка використовується для виконання команд через оболонку та повернення повного результату у вигляді рядка. Ми будемо її використовувати для управління мережевим екраном на прикладному рівні.

Параметри: ця функція приймає один параметр $cmd , який використовується для зберігання команди, яка буде виконана.

Приклад використання: $output = shell_exec('ls');

3.3 Вибір вебсерверу

Для реалізації http-серверу був обраний Workerman - це асинхронний PHP-фреймворк, керований подіями, із високою продуктивністю для створення швидких і масштабованих мережевих програм. Workerman підтримує HTTP, Websocket, SSL та інші спеціальні протоколи. Підтримка протоколу Websocket дозволить налаштувати безперервний зв’язок між серверною та клієнтською частинами модуля.

Workerman PHP є потужним фреймворком для розробки серверних додатків на мові програмування PHP. Він забезпечує можливість створювати високопродуктивні сервери, які можуть обробляти багато одночасних з'єднань з великою швидкістю.

Основні особливості та переваги Workerman PHP:

- Асинхронна модель програмування: Workerman використовує асинхронний підхід, що дозволяє одночасно обробляти багато з'єднань без блокування і переключення між ними. Це робить його ефективним у роботі з багатопоточними та багатозадачними середовищами.
- Висока продуктивність: завдяки асинхронності і ефективному використанню ресурсів, Workerman забезпечує високу продуктивність та швидкість обробки запитів.
- Підтримка протоколів: Workerman підтримує різні мережеві протоколи, такі як HTTP, WebSocket, TCP, UDP і Unix-сокети. Це дозволяє легко розробляти різноманітні серверні додатки залежно від потреб проекту.
- Гнучкість і розширюваність: Workerman надає багато функцій та інструментів для налаштування та розширення. Ми можемо створювати власні компоненти та розширення, щоб пристосувати його до своїх потреб.
- Простота використання: Workerman має простий та зрозумілий синтаксис, що дозволяє швидко розпочати розробку серверних додатків. Ми можете швидко побудувати масштабовані і надійні додатки з мінімальними зусиллями.

Загалом, Workerman PHP є потужним інструментом для розробки серверних додатків на мові програмування PHP, який дозволяє створювати високопродуктивні та масштабовані додатки зі зручним синтаксисом та багатими можливостями. Завдяки асинхронній моделі, даний інструмент дозволяє ефективно обробляти багато одночасних з'єднань без блокування виконання інших операцій. Це забезпечує швидку та ефективну роботу з мережевими додатками. Даний фреймворк має активну спільноту розробників, що сприяє обміну знаннями, підтримці та розвитку фреймворку. Розробники можуть швидко знайти документацію, приклади коду і отримати відповіді на свої питання.

Приклади використання:

- Сервер http, рисунок 1.4.

```php
use Workerman\Worker;
require_once __DIR__ . '/vendor/autoload.php';
$http_worker = new Worker('http://0.0.0.0:2345');

// Видається під час отримання даних
$http_worker->onMessage = function ($connection, $request) {
    //$request->cookie();
    //$request->path();
    // Надіслати дані клієнту
    $connection->send("Hello World");
};

// Запустити всі сервери
Worker::runAll();
```

Рисунок 1.4 – код http-сервера

- Сервер websocket, рисунок 1.5.

```php
use Workerman\Worker;
require_once __DIR__ . '/vendor/autoload.php';
// Створення серверу Websocket
$ws_worker = new Worker('websocket://0.0.0.0:2346');
// Видається, коли виникає нове підключення
$ws_worker->onConnect = function ($connection) {
    echo "New connection\n";
};
// Видається під час отримання даних
$ws_worker->onMessage = function ($connection, $data) {
    // Send hello $data
    $connection->send('Hello ' . $data);
};
// Видається, коли з'єднання закрито
$ws_worker->onClose = function ($connection) {
    echo "Connection closed\n";
};
// Запустити всі сервери
Worker::runAll();
```

Рисунок 1.5 – код ws-сервера

- Таймер, рисунок 1.6.

```php
use Workerman\Timer;
require_once __DIR__ . '/vendor/autoload.php';
//таймер на 5 секунд
$sec = 5;
$timer_id = Timer::add($sec, function () use (&$timer_id) {
    //виконання коду
    Timer::del($timer_id);
});
```

Рисунок 1.6 – код таймера

Дані можливості допоможуть нам реалізувати програмний модуль забезпечення захисту вебзастосунків від DDoS-атак. Також в ході розробки буде створений інтерфейс спостереження, завдяки якому адміністратор зможе швидше реагувати на наявні атаки.

3.4 Розробка серверної частини

Після того, як був розроблений алгоритм ідентифікації DDoS-атаки, механізм реагування на дані атаки та вибраний вебсервер для виконання поставленої задачі, прийшов час розробки програмного модуля для підвищення захисту від атак типу «відмова в обслуговуванні».

Розпочати даний процес необхідно з визначення конфігураційних змінних, в яких будуть зберігатися налаштування модуля:

- $localsocket – URL для прослуховування tcp-сервером. Даний сервер потрібен для передачі даних від процесу http-сервера до ws-сервера.
- $websocket – URL для прослуховування ws-сервером. Даний сервер потрібен для встановлення з’єднання між клієнтом і сервером. Дане з’єднання необхідне для відправки інформації до адміністратора.
- $token – секретний токен для ідентифікації, аутентифікації та авторизації ws-клієнта.
- $users – масив з об’єктами підключення адміністраторів.
- $ip_list – двовимірний масив з часовими мітками http-запитів.
- $time_block – кількість секунд, на яку необхідно заблокувати IP-адресу.
- $max_request – максимальна кількість запитів для переходу до наступної умови.
- $time_interval – мінімальна кількість секунд, протягом яких можна зробити $max_request запитів.

Визначення цих змінних зображено на рисунку 1.7.

```
$localsocket = "tcp://127.0.0.1:1111";
$websocket = "websocket://0.0.0.0:4000";
$token = "vEy1CMwtM5iVYi8l4UANlNQj1x2h1oUjxfZYJ4PlyhSMwIDA";
$users = [];
$ip_list = [];
$time_block = 60;
$max_request = 10;
$time_interval = 10;
```

Рисунок 1.7 – параметри модуля

Далі потрібно розробити функцію, в яку http-сервер буде передавати IP-адресу джерела запиту та метод запиту. В ній буде реалізований алгоритм ідентифікації DDoS-атаки та застосований механізм реагування на атаку. Її реалізацію зображено на рисунку 1.8.

```
function check_ip($ip, $method)
{
    $time_now = time();
    $GLOBALS['ip_list'][$ip][] = $time_now;
    $arr_time = $GLOBALS['ip_list'][$ip];
    send_to_front(["code" => 1, "ip" => $ip, "method" => $method]);
    if (count($arr_time) > $GLOBALS['max_request']) {
        $arr_time = array_slice($arr_time, -$GLOBALS['max_request']);
        if (end($arr_time) - $arr_time[0] <= $GLOBALS['time_interval']) block_ip($ip);
    }
}
```

Рисунок 1.8 – функція перевірки IP

Після ідентифікації однієї з IP-адрес ботнету, який проводить атаку, необхідно її заблокувати. Це реалізовано шляхом добавляння правила блокування вхідного трафіку. Для запуску команд блокування та розблокування було використано раніше описану функцію shell_exec, а для того, щоб команда виконалась у фоновому режимі, потік виводу був направлений у /dev/null. Також потрібно створити відкладену задачу на розбокування даної адреси.

Приклад цієї реалізації зображено на рисунку 1.9.

```php
function block_ip($ip)
{
    $time_unlock = time() + $GLOBALS['time_block'];
    unset($GLOBALS['ip_list'][$ip]);
    shell_exec("sudo iptables -A INPUT -s {$ip} -j DROP > /dev/null 2>&1 &");
    send_to_front(["code" => 2, "ip" => $ip, "time" => $time_unlock]);
    $timer_id = Timer::add($GLOBALS['time_block'], function () use (&$timer_id, $ip) {
        shell_exec("sudo iptables -D INPUT -s {$ip} -j DROP > /dev/null 2>&1 &");
        send_to_front(["code" => 3, "ip" => $ip]);
        Timer::del($timer_id);
    });
}
```

Рисунок 1.9 – функція блокування IP

Для відправки інформації в панель адміністратора необхідно налаштувати вебсокет сервер та локальний tcp-сервер, який слугуватиме мостом між процесами http-сервера та ws-сервера. Налаштування сервера зображено на рисунку 2.1.

```php
$ws_worker->onWorkerStart = function () use (&$users) {
    $inner_tcp_worker = new Worker($GLOBALS['localsocket']);
    $inner_tcp_worker->onMessage = function ($connection, $data) use (&$users) {
        foreach ($users as $var) {
            $webconnection = $var;
            $webconnection->send($data);
        }
    };
    $inner_tcp_worker->listen();
};
$ws_worker->onConnect = function ($connection) use (&$users) {
    $connection->onWebSocketConnect = function ($connection) use (&$users) {
        if (empty($_GET['token'])) return false;
        else {
            $token = trim($_GET['token']);
            if ($token == $GLOBALS['token'])$users[] = $connection;
            else return false;
        }
    };
};
$ws_worker->onClose = function ($connection) use (&$users) {
    $user = array_search($connection, $users);
    if ($user !== false) unset($users[$user]);
};
```

Рисунок 2.1 – налаштування вебсокет сервера

Дані обробники подій будуть реалізовувати підключення та відключення клієнта, а також відправку даних.

Після реалізації попередніх кроків потрібно запустити скрипт в режимі демона(опція -d). Запуск модуля зображено на рисунку 2.2

```
anton@247bf3107850:/var/www/html$ php index.php start -d
Workerman[index.php] start in DAEMON mode
------------------------------------------ WORKERMAN ------------------------------------------
Workerman version:4.0.19          PHP version:7.3.31-1~deb10u3
------------------------------------------ WORKERS -------------------------------------------
proto   user            worker          listen                      processes   status
tcp     anton           WS Server       websocket://0.0.0.0:4000    1           [OK]
tcp     anton           HTTP Server     http://0.0.0.0:80           1           [OK]
----------------------------------------------------------------------------------------------
Input "php index.php stop" to stop. Start success.
```

Рисунок 2.2 – запуск програмного модуля

3.5 Розробка інтерфейсу для спостереження

Розробка інтерфейсу для спостереження на фронтенді вимагає планування, дизайну та реалізації. Вебсторінка буде представляти з себе дві таблиці, які будуть оновлюватись в реальному часі.

Перша таблиця – це поточні http-запити. Кожен запит на сервер буде відправлятися адміністратору в панель та матиме три параметри: IP-адреса джерела, час отримання запиту та метод запиту(GET, POST, HEAD тощо).

Друга таблиця – це заблоковані IP-адреси або ж правила мережевого екрана для блокування певних адрес. Вона містить три поля: IP-адреса заблокованого джерела, час створення правила блокування і таймер, який відлічує час до видалення даного правила.

Використовуючи наведені вище потреби, було спроектовано наступну сторінку, рисунок 2.3.

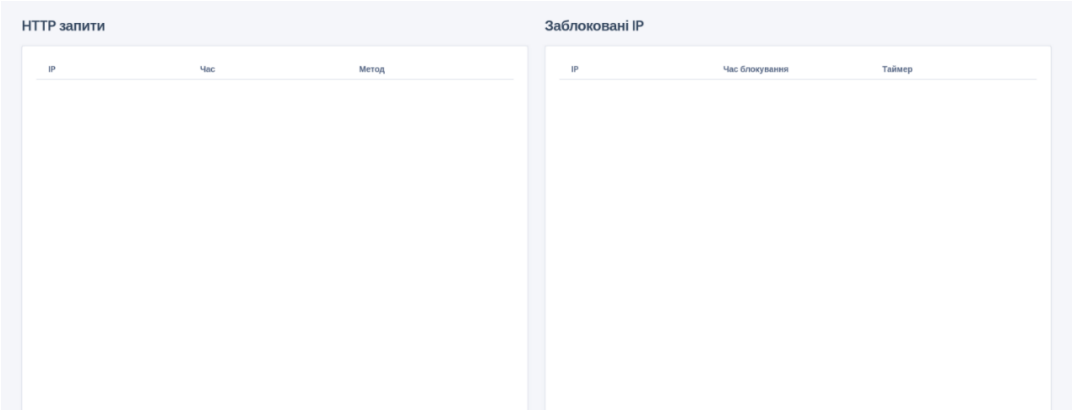

Рисунок 2.3 – шаблон панелі

Це лише макет сторінки, який не підкріплений функціональними можливостями. Для отримання даних в реальному часі необхідно використовувати JavaScript - це високорівневнева мова програмування, що широко використовується для розробки динамічних та інтерактивних вебсторінок. Вона працює на більшості браузерів і використовується як мова скрипту для взаємодії з користувачем, маніпулювання DOM (Document Object Model), виконання асинхронних запитів до сервера та іншого функціоналу.

Для спрощення роботи з DOM буде використано jQuery - це популярна бібліотека JavaScript, яка спрощує маніпулювання DOM, роботу з подіями, анімаціями та взаємодію з AJAX на вебсторінках. Вона надає багато готових функцій і методів, що полегшують розробку вебдодатків і скорочують код.

Встановити постійне з’єднання з сервером допоможе об’єкт JS WebSocket - це API, яке надає можливість встановлювати двосторонні з'єднання між браузером та сервером через протокол WebSocket. Воно дозволяє передавати дані між клієнтом і сервером в реальному часі.

Основні етапи роботи з JS WebSocket:

1. Встановлення з'єднання: спочатку потрібно створити WebSocket-об'єкт на стороні клієнта і вказати URL сервера, з яким потрібно встановити з'єднання. Це зображено на рисунку 2.4.

```
const socket = new WebSocket('ws://example.com/socket');
```

Рисунок 2.4 – створення WebSocket-об'єкта

2. Обробка подій: WebSocket надсилає різні події, які можна прослуховувати, щоб реагувати на зміни стану з'єднання, отримання, відправлення повідомлень. Найпоширеніші події – open, message, error та close. Приклади обробки даних подій зображено на рисунку 2.5.

```
socket.addEventListener('open', () => {
    console.log(`З'єднання встановлено`);
});
socket.addEventListener('message', (event) => {
    var message = event.data;
    console.log('Отримано повідомлення:', message);
});
socket.addEventListener('error', (error) => {
    console.error(`Помилка з'єднання:`, error);
});
socket.addEventListener('close', () => {
    console.log(`З'єднання закрито`);
});
```

Рисунок 2.5 – слухачі та обробники подій

3. Надсилання та отримання повідомлень: потрібно використовувати метод send() WebSocket-об'єкта для відправки повідомлень на сервер. На сервері можна обробити ці повідомлення і відправити відповідь клієнту. Реалізацію даного метода зображено на рисунку 2.6.

```
socket.send('Запит до сервера');
socket.addEventListener('message', (event) => {
  var message = event.data;
  console.log('Отримано повідомлення:', message);
});
```

Рисунок 2.6 – надсилання та отримання даних

4. Закриття з'єднання: щоб закрити з'єднання WebSocket, необхідно використати метод close(). Це зображено на рисунку 2.7.

```
socket.close();
```

Рисунок 2.7 – закриття сокету

JavaScript WebSocket дозволяє взаємодіяти з сервером у реальному часі, передавати дані та отримувати оновлення без необхідності оновлювати веб-сторінку. Це корисний інструмент для створення додатків, які потребують взаємодії між

клієнтом та сервером в реальному часі, таких як чати, спостережні панелі та багатокористувацькі ігри.

3.6 Перевірка роботи програмного модуля

Для проведення перевірки роботи необхідно використовувати різні IP-адреси джерел запитів. Цю можливість надає мережа Tor – це відкрита мережа, що допомагає захистити анонімність в Інтернеті. Це робиться шляхом направлення з'єднань через серію віртуальних тунелів, замість безпосереднього підключення, що забезпечує кращу анонімність.

Трафік в мережі Tor проходить через три випадково обрані вузли: вхідний вузол, проміжний вузол і вихідний вузол. Кожен вузол знає лише про вузол, який перед ним, і вузол, який слід за ним, але не знає всього шляху. Це допомагає захистити анонімність користувача.

Спочатку зробимо декілька запитів для перевірки таблиці з http-запитами. Результат виявився вдалим, рисунок 2.8.

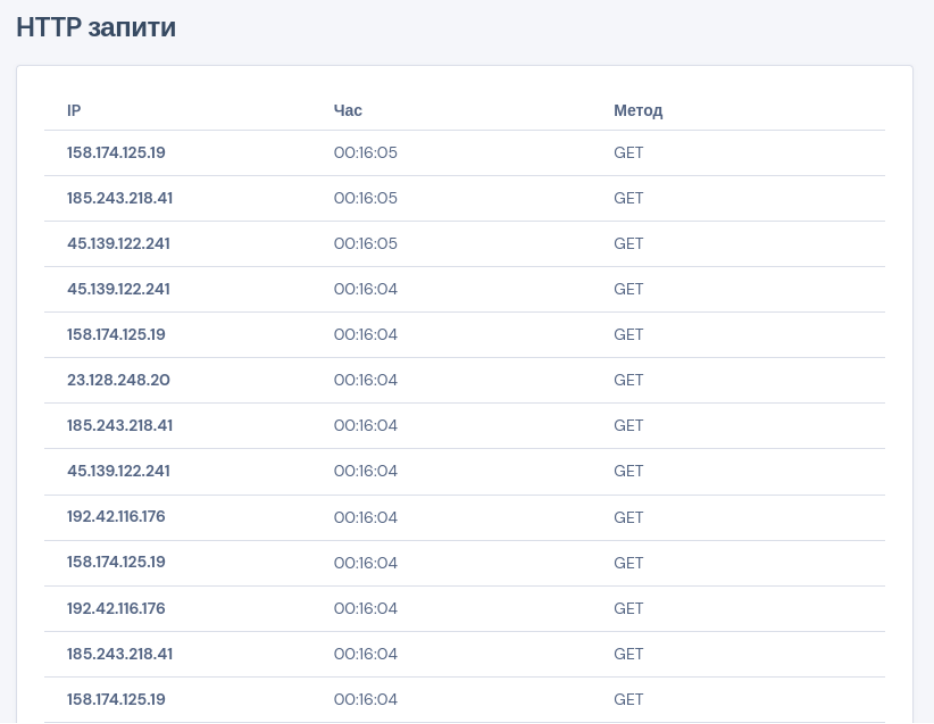

Рисунок 2.8 – список http-запитів

Далі необхідно реалізувати DDoS-атаку для перевірки працездатності модуля. Було використано п’ять IP-адрес з мережі Tor. Після недовготривалої атаки дані адреси були добавлені в правила блокування. Можна помітити, що час блокування кожної IP-адреси збігається з часом останнього запиту від неї. Результат зображено на рисунку 2.9.

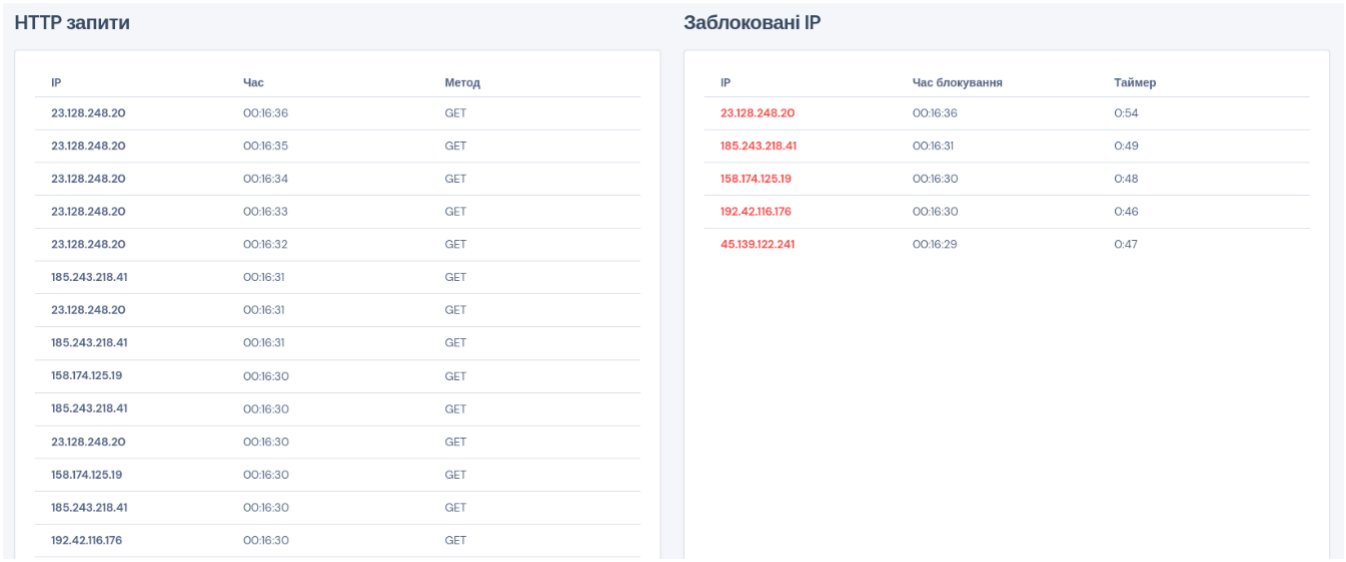

Рисунок 2.9 – результат роботи модуля

Даний результат говорить про успішну роботу модуля. Перевірити список правил фаєрвола можна за допомогою команди iptables -L INPUT -v -n. Вона виведе нам список правил, в яких вказано заблоковані IP-адреси, це допоможе звіритись з фронтендом. Результат виконання даної команди зображено на рисунку 3.1.

```
anton@247bf3107850:/var/www/html$ sudo iptables -L INPUT -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   31  1816 DROP       all  --  *      *       185.220.101.7        0.0.0.0/0
   22  1048 DROP       all  --  *      *       109.70.100.16        0.0.0.0/0
   30  1452 DROP       all  --  *      *       185.220.101.75       0.0.0.0/0
   20   944 DROP       all  --  *      *       185.220.102.250      0.0.0.0/0
   25  1460 DROP       all  --  *      *       71.19.144.89         0.0.0.0/0
```

Рисунок 3.1 – правила фаєрвола

3.7 Переваги та недоліки

Програмний модуль для захисту від DDoS-атак має ряд переваг:

- Специфічність: розроблений модуль може бути спеціально налаштований на ваші вимоги і зосереджений на конкретних типах DDoS-атак, які ваша мережа чи система найчастіше відбиває.
- Гнучкість: програмний модуль може бути гнучким і адаптованим до змін у вашій інфраструктурі, політиках безпеки та вимогах до служби.
- Інтеграція: власний модуль може бути інтегрований безпосередньо у ваші системи, що дозволяє краще координувати заходи безпеки і реагування на інциденти.
- Вартість: хоча вартість розробки та підтримки може бути високою, ви можете заощадити на вартості використання послуг третьої сторони, якщо у вас достатньо високий обсяг або специфічні потреби.
- Контроль: власний модуль надає вам повний контроль над своїми заходами з мережевої безпеки і відповіддю на DDoS-атаки.

Хоча програмний модуль має багато переваг, є й певні недоліки, які варто врахувати:

- Складність розробки: створення власного програмного рішення для захисту від DDoS може бути технічно складним завданням, що вимагає великої кількості ресурсів та експертизи.
- Вартість: навіть якщо ви заощадите на вартості послуг третьої сторони, вартість розробки, впровадження та підтримки власного рішення може бути високою.
- Підтримка та оновлення: оскільки загрози DDoS постійно розвиваються, програмний модуль має постійно оновлюватись, що вимагає постійних вкладень та підтримки.
- Масштабованість: більшість власних рішень важко масштабувати, особливо в порівнянні з рішеннями, що надаються великими провайдерами.
- Ризик помилок: є ризик, що програмне рішення може мати баги або вразливості, які можуть бути використані атакуючими.

Також варто відмітити кілька залежностей, які присутні у даного модуля:

- Операційна система: Linux.
- Вебсервер: Workerman або інший сервер на PHP.
- Мережевий екран: Iptables.
- Мова програмування: PHP.

Висновки за розділом 3

Створення програмного модуля для захисту від DDoS-атак - це великий і складний процес, який вимагає глибокого розуміння мережевих протоколів, алгоритмів ідентифікації та міграції атак.

Під час виконання даної роботи було сформульовано декілька основних кроків, які б допомогли створити подібний або більш досконалий модуль:

- Аналіз трафіку: першим кроком є аналіз трафіку, щоб ідентифікувати нормальні шаблони та відхилення, які можуть вказувати на DDoS-атаку. Це може включати в себе аналіз різних параметрів, таких як швидкість передачі даних, кількість пакетів від певного джерела, частоту нових з'єднань та інше.
- Ідентифікація атак: наступним кроком є використання цих даних для ідентифікації потенційних DDoS-атак. Це може включати в себе створення алгоритмів машинного навчання для визначення аномалій або використання вже відомих шаблонів атак.
- Зм'якшення атак: після ідентифікації атаки потрібно вжити заходів для її зм'якшення. Це може включати в себе блокування IP-адрес джерела атаки, обмеження швидкості трафіку, редирект трафіку через спеціальну мережу (наприклад, через CDN або спеціальну мережу для зм'якшення DDoS) або зміну конфігурації сервера для зниження впливу атаки.
- Відновлення після атаки: після того, як атака була зупинена, важливо відновити нормальне функціонування системи і проаналізувати атаку, щоб зрозуміти, як можна було б запобігти подібним атакам у майбутньому.

Цей процес вимагає великого обсягу знань і досвіду в області мережевої безпеки, а також значного вкладу часу та ресурсів. Тому для багатьох організацій може бути набагато ефективніше використовувати існуючі рішення для захисту від DDoS-атак, такі як Cloudflare, Akamai, AWS Shield та інші.

ВИСНОВКИ

DDoS-атаки представляють серйозну загрозу для організацій усіх видів, оскільки вони можуть спричинити значні збої в роботі систем та сервісів, викликаючи втрату бізнесу, недовіру клієнтів та інші негативні наслідки. Вони здатні перевантажити ваші сервери та мережеву інфраструктуру, спричиняючи їхню недоступність для користувачів. Це може призвести до серйозних перерв у роботі вашого бізнесу. Втрати від перерв у роботі можуть бути значними, особливо для організацій, які залежать від онлайн-бізнесу. Додатково, ви можете витратити значні ресурси на відновлення від атаки. Варто зазначити, що дані атаки можуть бути використані як відволікання від інших типів атак. Наприклад, кіберзлочинці можуть запустити DDoS-атаку, щоб відвернути увагу від спроби проникнення в систему. Деякі організації, зокрема у фінансовій та медичній галузях, зобов'язані законодавством або нормативними вимогами захищати свої системи від DDoS-атак.

Насамперед було розглянуто визначення атаки типу «відмова в обслуговуванні», та наведено приклад їх застосування. Для кращого розуміння природи цих атак, була наведена коротка історія їх розвитку. Знаючи дані подробиці, було проведено аналіз видів DDoS-атак та розглянуто методи захисту від даних атак. Були виділені характерні риси найпопулярніших видів атак, таких як: атаки на рівні мереж, атаки на рівні транспортного протоколу, атаки на рівні застосунків, ампліфікаційні атаки та синхронізовані атаки.

Виконання даних задач повністю вирішує завдання, яке поставлене у першому розділі.

Після вивчення теоретичних матеріалів та проведення, були розроблені та наведені рекомендації щодо подальшої розробки програмного модуля. Для початку був проведений аналіз загроз, які існують при DDoS-атаці. Це дало поштовх до розгляду методів, які потрібно реалізувати у коді. Також були розглянуті загальні принципи забезпечення захисту на рівні програмного забезпечення.

Виконання даних робіт, призвело до того, що завдання другого розділу було виконане.

Виконавши завдання першого та другого розділів, настав час перейти до практичної частини роботи. Завдяки попередньо отриманим навичкам був розроблений алгоритм ідентифікації DDoS-атак та спосіб реагування на дану атаку. Далі був підібраний вебсервер, який забезпечує надійну роботу та має достатню документацію. Завдяки цим складовим вдалось розробити серверну та клієнтські частини програмного модуля. Також наведено його основні переваги, недоліки та залежності.

Виконавши на практиці попередньо описані кроки, було виконане завдання третього розділу дипломної роботи. В даній роботі був побудований програмний модуль забезпечення захисту вебзастосунків від DDoS-атак, а також розроблено вебпанель для спостереження.

Виходячи із поставленої мети дипломної роботи були виконані наступні завдання:

- Розглянуто та проаналізовано типи DDoS-атак.
- Досліджено основні методи захисту від даних атак.
- Виведено рекомендації для розробки програмного забезпечення.
- Створено алгоритм ідентифікації DDoS-атак.
- Розроблено та описано скрипт на мові PHP для ідентифікації та реагування на атаки.
- Побудована клієнтська частина для спостереження, яка відображає інформацію в реальному часі.
- Наведено основні переваги, недоліки та залежності даного програмного модуля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance [Електронний ресурс] / K. B. Dhruba, K. K. Jugal. – 2015. – Режим доступу до ресурсу: https://discovery.hw.ac.uk/primo-explore/fulldisplay/44hwa_alma2150137990003206/44HWA_V1.

2. DDoS Attacks: Effective Detection and Defense / S. Gupta, R. Kumar.

3. The distributed denial of service attacks (DDoS) prevention mechanisms on application layer [Електронний ресурс] / S. Bhosale, M. Nenova, G. Iliev. – 2017. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/8246247.

4. Internet Denial of Service: Attack and Defense Mechanisms [Електронний ресурс] / J.Mirkovic, S. Dietrich, D. Dittrich, P. Reiher. – 2004. – Режим доступу до ресурсу: https://dl.acm.org/doi/book/10.5555/1044905.

5. What is a DNS amplification attack [Електронний ресурс] – Режим доступу: https://www.imperva.com/learn/application-security/dns-amplification/.

6. DDoS: Threat Landscape and Best Practices for Prevention and Mitigation / Radware.

7. The CERT Guide to System and Network Security Practices [Електронний ресурс] / Julia Allen. – 2001. – Режим доступу до ресурсу: https://www.researchgate.net/publication/247804285_The_CERT_Guide_to_System_and_Network_Security_Practices.

8. DDoS Mitigation: Techniques, Tools, and Technologies / A. Nazario.

9. The Practical Guide to Information Security / Alex Holden.

10. Distributed Denial of Service Attack and Defense [Електронний ресурс] / Shui Yu. – 2013. – Режим доступу до ресурсу: https://dl.acm.org/doi/10.5555/2556385.

11. Distributed denial of service attacks in software / Q. Yan, R. Yu.

12. Cyber Operations: Building, Defending, and Attacking Modern Computer Networks [Електронний ресурс] / Mike O'Leary. – 2015. – Режим доступу до ресурсу:

https://discovery.hw.ac.uk/primo-explore/fulldisplay/44hwa_alma2150137990003206/44HWA_V1.

13. DDoS Attacks in Cloud Computing: Issues, Taxonomy, and Future Directions [Електронний ресурс] / G. Somani, D. Sanghi, M. Conti. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/288713585_DDoS_Attacks_in_Cloud_Computing_Issues_Taxonomy_and_Future_Directions.

14. Cybersecurity: Protecting Critical Infrastructures from Cyber Attack and Cyber Warfare [Електронний ресурс] / Thomas Johnson. – 2015. – Режим доступу до ресурсу: https://www.taylorfrancis.com/books/edit/10.1201/b18335/cybersecurity-thomas-johnson.

15. Network Security A Beginner's Guide, Third Edition [Електронний ресурс] / Eric Maiwald. – 2013. – Режим доступу до ресурсу: https://www.mheducation.com/highered/product/network-security-beginner-s-guide-third-edition-maiwald/9780071795708.html.

16. Performance Evaluation of AODV Protocol under DDoS Attacks in MANET [Електронний ресурс] / S. Saraeian – 2008. – Режим доступу до ресурсу: https://www.researchgate.net/publication/331653814_Performance_Evaluation_of_AODV_Protocol_under_DDoS_Attacks_in_MANET.

17. A survey on methods to defend against DDoS / S.Farahmandian, M. Zamani, A. Akbarabadi, M. Zadeh. – 2013.

18. Preserving DDoS Attacks Using Node Blocking Algorithm [Електронний ресурс] / R. Punidha, K. Pavithra, R. Swathika. – 2019. – Режим доступу до ресурсу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3352101.

19. Preventing Distributed Denial-of-Service Flooding Attacks With Dynamic Path Identifiers [Електронний ресурс] / Z. Chen, L. Jiawei, A. Vasilakos. – 2017. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/7888484.

20. A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method [Електронний ресурс] / P. Bojović, I. Bašičević, S. Ocovaj. –

2018. – Режим доступу до ресурсу: https://www.semanticscholar.org/paper/A-practical-approach-to-detection-of-distributed-a-Bojovi%C4%87-Basicevic/840e888413245b56ad237c5735626dbf497eafed.

21. Cybersecurity and Cyberwar: What Everyone Needs to Know [Електронний ресурс] / P. Singer, A. Friedman. – 2014. – Режим доступу до ресурсу: https://books.google.com.ua/books/about/Cybersecurity.html?id=9VDSAQAAQBAJ.

22. Famous DDoS Attacks | The Largest DDoS Attacks Of All Time [Електронний ресурс] – Режим доступу: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/.

23. Linux Firewalls: Enhancing Security with nftables and Beyond [Електронний ресурс] / Steve Suehring. – 2015. – Режим доступу до ресурсу: https://el.newoutlook.it/download/book/Linux-Firewalls-Enhancing-Security-with-nftables-and-Beyond.pdf.

24. Building Internet Firewalls [Електронний ресурс] / E. Zwicky, S. Cooper, D. Chapman. – 2000. – Режим доступу до ресурсу: https://www.oreilly.com/library/view/building-internet-firewalls/1565928717/.

25. Lightweight DDoS flooding attack [Електронний ресурс] / R. Braga, E. Mota, A. Passito. – 2011. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/5735752.

26. Перерахування та видалення правил брендмауера Iptables" [Електронний ресурс] – Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-list-and-delete-iptables-firewall-rules-ru.

27. Посібник PHP [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/index.php.

28. Документація Workerman [Електронний ресурс] – Режим доступу: https://www.workerman.net/doc/workerman/ .

29. Посібник JavaScript [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

30. Документація Jquery [Електронний ресурс] – Режим доступу: https://jquery.com/ .

ДОДАТОК А
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Шпильовий А.М., Фесенко А.О. Використання Cloudflare для підвищення захисту від DDoS-атак. Науково-практична конференція «ПРОБЛЕМИ ЕКСПЛУАТАЦІЇ ТА ЗАХИСТУ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ».
2. Шпильовий А.М., Фесенко А.О. OSSEC для виявлення DDoS-атак. Науково-практична конференція «ПРОБЛЕМИ ЕКСПЛУАТАЦІЇ ТА ЗАХИСТУ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ».

ДОДАТОК Б
КОД СЕРВЕРНОЇ ЧАСТИНИ

```
use Workerman\Worker;
use Workerman\Timer;
require "conf.php";
$ws_worker = new Worker($websocket);
$ws_worker->name = 'WS Server';
function check_ip($ip, $method)
{
    $time_now = time();
    $GLOBALS['ip_list'][$ip][] = $time_now;
    $arr_time = $GLOBALS['ip_list'][$ip];
    send_to_front(["code" => 1, "ip" => $ip, "method" => $method]);
    if (count($arr_time) > $GLOBALS['max_request']) {
        $arr_time = array_slice($arr_time, -$GLOBALS['max_request']);
        if (end($arr_time) - $arr_time[0] <= $GLOBALS['time_interval']) block_ip($ip);
    }
}
function block_ip($ip)
{
    $time_unlock = time() + $GLOBALS['time_block'];
    unset($GLOBALS['ip_list'][$ip]);
    shell_exec("sudo iptables -A INPUT -s {$ip} -j DROP > /dev/null 2>&1 &");
    send_to_front(["code" => 2, "ip" => $ip, "time" => $time_unlock]);
    $timer_id = Timer::add($GLOBALS['time_block'], function () use (&$timer_id, $ip) {
        shell_exec("sudo iptables -D INPUT -s {$ip} -j DROP > /dev/null 2>&1 &");
        send_to_front(["code" => 3, "ip" => $ip]);
        Timer::del($timer_id);
    });
```

```
}
function send_to_front($arr)
{
   $instance = stream_socket_client($GLOBALS['localsocket']);
   fwrite($instance, json_encode($arr)  . "\n");
}
$ws_worker->onWorkerStart = function () use (&$users) {
   $inner_tcp_worker = new Worker($GLOBALS['localsocket']);
   $inner_tcp_worker->onMessage = function ($connection, $data) use (&$users) {
      foreach ($users as $var) {
         $webconnection = $var;
         $webconnection->send($data);
      }
   };
   $inner_tcp_worker->listen();
};
$ws_worker->onConnect = function ($connection) use (&$users) {
   $connection->onWebSocketConnect = function ($connection) use (&$users) {
      if (empty($_GET['token'])) return false;
      else {
         $token = trim($_GET['token']);
         if ($token == $GLOBALS['token'])$users[] = $connection;
         else return false;
      }
   };
};
$ws_worker->onClose = function ($connection) use (&$users) {
   $user = array_search($connection, $users);
   if ($user !== false) unset($users[$user]);};
```

ДОДАТОК В
КОД КЛІЄНТСЬКОЇ ЧАСТИНИ

```
var ws = new WebSocket(ws_url);
var all_ip = $(".all_ip_table>tbody"),
    blocked_ip = $(".blocked_table>tbody");
setTimeout(() => {
    setInterval(() => {
        if (ws.readyState == 3) {
            ws = new WebSocket(ws_url);
            ws.onmessage = function(e) { message_socket(e); };
        }
    }, 1000);
}, 3000);
ws.onmessage = function(e) {
    message_socket(e);
};
function message_socket(e) {
    var arr = JSON.parse(e.data);
    var code = parseInt(arr['code']);
    switch (code) {
        case 1:
            all_ip.prepend(`<tr><th scope="row">${arr.ip}</th><td>${(new
Date()).toLocaleTimeString()}</td><td>${arr.method}</td></tr>`);
            break;
        case 2:
            blocked_ip.prepend(`<tr data-ip='${arr.ip}'><th
scope="row">${arr.ip}</th><td>${(new Date()).toLocaleTimeString()}</td><td
class='timer'>???</td></tr>`);
            blocked_timer(arr.ip, arr.time);
```

```
            break;
        case 3:
            blocked_ip.find(`tr[data-ip='${arr.ip}']`).remove();
            break;
        default:
            break;
    }
};

function blocked_timer(ip, time) {
    var all_sec = time - parseInt(Date.now() / 1000);
    var timer_td = blocked_ip.find(`tr[data-ip='${ip}'] .timer`);
    var timer = setInterval(() => {
        var mins = parseInt(all_sec / 60);
        var sec = (all_sec % 60).toString().padStart(2, '0');
        timer_td.text(`${mins}:${sec}`);
        all_sec--;
        if (all_sec < 0) {
            clearInterval(timer);
        }
    }, 1000);
}
```