Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи протидії DDoS-атакам у комп’ютерній мережі”

КбПЗ_2025

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Герман Д.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Герману Дмитру Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи протидії DDoS-атакам у комп’ютерній мережі*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 48-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи протидії DDoS-атакам у комп’ютерній мережі*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

КПІ 2025

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

КбПЗ_2025

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Герман Д.В.
(прізвище та ініціали)

АНОТАЦІЯ

Герман Д.В. Програмне забезпечення системи протидії DDoS-атакам у комп'ютерній мережі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи протидії DDoS-атакам у комп'ютерній мережі.

Метою розробки є програмне забезпечення системи протидії DDoS-атакам у комп'ютерній мережі.

Результат роботи – програмна реалізація системи протидії DDoS-атакам у комп'ютерній мережі.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, DDoS-атака, комп'ютерна мережа

КПЗ – 2025

ABSTRACT

German D.V. Software for the system of countering DDoS attacks in a computer network. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the system of countering DDoS attacks in a computer network.

The purpose of the development is software for the system of countering DDoS attacks in a computer network.

The result of the work is the software implementation of the system of countering DDoS attacks in a computer network.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, DDoS attack, computer network

КБП3_2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи..... 5
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 23
2.3 Розгорнута постановка завдання 27
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 29
3.1 Опис функціонування системи 29
3.2 Розробка структурної схеми..... 30
3.3 Розробка функціональної схеми 36
3.4 Розробка діаграми процесів..... 50
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 52
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 52
4.2 Захист розробленого програмного забезпечення..... 63
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 65
6 ОСНОВНІ ВИСНОВКИ..... 69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 71

КОПЗ – 2025

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
КВ	–	коефіцієнт варіації
КЗ	–	канал зв'язку
НСД	–	несанкціонований доступ
ПС	–	програмна середа
СВВ	–	система виявлення вторгнень
СеМО	–	експонентна мережа масового обслуговування
СМО	–	система масового обслуговування
СПД	–	система передачі даних

КБПЗ_2025

ВСТУП

Актуальність теми. Атаки в мережі Web не завжди пов'язані з одержанням несанкціонованого доступу до інформації. Розподілені атаки з метою викликати відмову в обслуговуванні (DDoS) наносять не меншу шкоду, і від них не зможуть захистити ні антивірусні сканери, ні брандмауери. Для боротьби з атаками DDoS потрібні спеціалізовані засоби захисту – бажано із хмари. На перші смуги газет регулярно попадають повідомлення про великомасштабні атаки Distributed Denial of Service (DDoS). У них утягуються десятки тисяч комп'ютерів одночасно, і вся ця сукупна ударна міць обрушується зловмисниками на обрану жертву. Спочатку розміщається троянець або бот на як можна більшому числі комп'ютерів або серверів, що мають широкополосне або виділене підключення до Інтернету. Як тільки центральний сервер ботнета (Command and Control Server, C&C Server) дає сигнал до початку атаки, всі задіяні комп'ютери одночасно відправляють запити обраному цільовому серверу, що приводить до його перевантаження. У гірших випадках атакована Web-сторінка може виявитися заблокованою на кілька днів, а особливо масові атаки здатні перекрити доступ навіть до ЦОД. Дуже часто безпомічними жертвами атак DDoS стають іменити компанії, які не подбали про підходящі засоби захисту. Серед потерпілих зустрічалися й такі, хто прийняв не саме вдале рішення при виборі апаратних засобів захисту від подібних атак, адже й у цій сфері продукти від різних постачальників значно розрізняються по якості. Іноді в них відсутні елементи, які інтернет-залежні підприємства визнали б, безсумнівно, украй важливими, якби знали про їхнє існування. При виборі підходящого рішення недостатньо покластися на голосне й популярне ім'я виробника – необхідно приділити пильну увагу деталям пропонованих рішень і розходженням між ними, щоб забезпечити найкращий захист своїх систем.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи протидії DDoS-атакам у комп’ютерній мережі.

КОПЗ–2025

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем протидії DDoS-атакам у комп’ютерній мережі.

– Дослідження системи протидії DDoS-атакам у комп’ютерній мережі.

– Програмна реалізація системи протидії DDoS-атакам у комп’ютерній мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі протидії DDoS-атакам у комп’ютерній мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи протидії DDoS-атакам у комп’ютерній мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2025

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Для IT-служби атака DDoS є серйозним випробуванням – потрібно виявити джерело атаки, з'ясувати її природу й виробити механізми захисту. При цьому засобу захисту повинні блокувати непродуктивні запити, ініційовані нападаючими й не компанії, що приносять, доходів.

Інтернет стає усе небезпечніше – на найбільш популярні ресурси всі частіше організуються атаки з метою викликати відмова в обслуговуванні (Denial of Service, DoS) і блокувати роботу окремих сайтів і цілих інформаційних систем. У цій статті ми постараємося розібратися в сучасних методах DoS-атак – точніше, розподілених DoS-атак (Distributed DoS, DDoS), які звичайно здійснюються зловмисниками за допомогою мережі зомбованих комп'ютерів.

Суть DDoS-атаки полягає в тому, що в інфраструктурі мережі жертви перебуває дефіцитний ресурс, вичерпання якого й викликає відмову в обслуговуванні. Найбільш відомі атаки недавнього часу були націлені на вичерпання пропускної здатності каналу підключення сайту до Інтернету. Однак розвиток широкополосних технологій доступу й хмарних обчислень ускладнює це завдання. Втім, зважаючи на все, труднощі зловмисників не лякають, і вони намагаються організувати усе могутніші атаки. Навесні цього року інфраструктура одного із хмарних провайдерів піддалася атаці потужністю 400 Гбіт/с, однак цілком можливі й більше масштабні акції.

Для їхньої організації використовують так званий метод посилення – додаткові підсилювачі замість прямого «закидання» серверів, що атакуються, пакетами, генеруємими на зомбованих комп'ютерах. Ідея підсилювача полягає в тому, що на якийсь проміжний сервіс підлегле зловмисникові пристрій посилає IP-пакет, у якому IP-адреса відправника замінена на адресу жертви. При цьому

КОПЗ-2025

вибираються такі сервіси, які у відповідь на дуже короткий запит повертають величезну кількість інформації. Наприклад, подібне поводження характерно для сервера DNS – у відповідь на запит синхронізації зон пересилається досить об'ємний зміст всієї зони.

Коефіцієнт підсилення – це відношення між розмірами запиту й потоку даних, що надходять від сервісу. Нападаючі намагаються знайти такі протоколи, які дозволяли б значно збільшувати потік. Зокрема, для цього підійдуть система дозволу імен DNS, служба синхронізації часу NTP, протокол мережного керування SNMP і деякі інші протоколи. Наприклад, згадана атака на хмарного провайдеру, що у піку мала потужність до 400 Гбіт/с, була організована за допомогою посилення на серверах NTP. На думку експертів різних лабораторій, гонка продовжиться – атакуючі почали вже захоплювати віртуальні машини в хмарах і з їхньою допомогою організовувати атаки по високошвидкісних каналах і з використанням всіх переваг хмар.

Втім, відзначається й інша тенденція: атаки проти ключових компонентів інфраструктури, таких як DNS-сервери. Дійсно, вивівши з ладу DNS, що обслуговує сайт, доступ до останнього можна перекрити. При цьому DNS-сервери виявилися менш захищеними, чим сайти, які розташовуються в хмарних середовищах. Оскільки у звичайній ситуації навантаження на DNS-сервер не дуже більша, цей елемент інфраструктури захищати від DDoS не було потрібно.

Болючою крапкою сайту можуть виявитися пристрою, установлені перед основним сервером: балансувальники навантаження, зворотні кеші й SSL-прискорювачі. Цілком можливо, що їхнє функціонування теж має певні особливості, які дозволять зловмисникам зробити сайт недоступним, не перевантажуючи канал непродуктивним навантаженням, з якого вже навчилися боротися спеціалізовані компанії.

Ще однією метою DoS-атак є виведення з ладу системи керування контентом сайту (Content Management System, CMS). Сучасні сайти є динамічними й будуються на основі системи керування контентом, що

КПЗ–2025

традиційно складається з Web-сервера, сервера додатків на основі якої-небудь сценарної мови й бази даних. Популярним зв'язуванням є LAMP – Linux, Apache, MySQL і PHP, є також стеки продуктів на основі Python, Ruby, Java, ASP і інших технологій. Найбільше поширення одержали безкоштовні CMS-продукти WordPress і Joomla, які базуються на стеці LAMP. Виводячи з ладу CMS, зловмисник блокує роботу сайту. Причому іноді це робиться за допомогою модулів розширення для CMS, куди вбудовується троянський компонент для дистанційного керування сайтом. Звичайно такий компонент використовується для зараження комп'ютерів відвідувачів сайту, але зловмисники можуть реалізувати й альтернативний метод атаки на власників, знищивши сайт зсередини.

Недавно з'явився новий тип DoS-атак – видалення даних у хмарних сервісах. Як правило, облікові дані від хмарних сервісів прив'язані до поштових адрес. Перехопивши або підібравши поштовий пароль, зловмисник може одержати контроль над хмарним сервісом і знищити збережені в ньому дані. Цим летом було зафіксовано дві атаки подібного типу. У першому випадку зловмисник перехопив облікових даних адміністраторів хмарного оператора Code Space і зажадав за них викуп. Адміністратори компанії спробували повернути контроль над своїми даними, і тоді атакуючий почав знищувати інформацію, що зберігається в хмарі. У результаті оператор був змушений оголосити, що вже не може надавати сервіс і звертає свою діяльність.

Трохи інша історія трапилася із британською компанією Aviva. До розгорнутого в неї продукту для керування мобільними пристроями MobileIron були прив'язані пристрої всіх співробітників. Зловмисник одержав доступ до облікових даним адміністратора системи, розіслав всім співробітникам SMS, а потім запустив вилучене обнуління даних на всіх корпоративних пристроях. Такі шкідливі дії теж можна віднести до DoS-атак, хоча для них є своя назва – саботаж. На щастя, поки зафіксовані тільки одиничні випадки саботажу й атак проти CMS. Вони явно організуються вручну. Оскільки автоматизованих

КОПЗ-2025

платформ для таких дій ще не створено, поки немає й потреби в автоматизованих засобах захисту від них. Проте при реалізації Web-проектів варто оцінювати ймовірність і такий тип атак.

1.2 Область застосування

Улітку поточного року відразу кілька компаній, які займаються захистом від DDoS-атак, опублікували статистику по традиційних атаках із залученням зомбі-комп'ютерів, підсилювачів і інших технічних хитрувань. Так, за даними компанії Prolexic, що проаналізувала DDoS-активність в II кварталі, середня потужність атак збільшилася на 72%, а пікова – на 241% у порівнянні з минулим роком. Правда, у порівнянні з I кварталом, коли була зафіксована максимальна пікова атака в 400 Гбіт/с, активність DDoS знизилася, при цьому скоротилася середня тривалість атак. Зараз вона становить 17 годин, чого, втім, цілком достатньо для нанесення досить серйозного збитку Web-проектам – як мінімум будуть потрібні чималі витрати на відновлення працездатності сайтів і придбання послуг із захисту від DDoS.

Arbor Networks відзначає, що в II кварталі кількість невеликих атак потужністю 20 Гбіт/с збільшилося вдвічі в порівнянні з аналогічним періодом минулого року. При цьому зафіксовано біля сотні атак потужністю понад 100 Гбіт/с. Ці виводи зроблені на підставі аналізу даних, що збираються в реальному часі спеціальною розподіленою системою, що здійснює моніторинг мережних подій в інфраструктурах 300 клієнтів компанії.

У II кварталі число атак зменшилося – з 4375 торік до 2715 у поточному, причому значно зросла кількість задіяних у них зомбі-комп'ютерів – з 136 644 до 420 489 машин у середньому на атаку. Крім того, збільшилася максимальна тривалість атак – з 21 до 90 днів. Експерти Qrator зафіксували зменшення частки атак з використанням методу посилення – цей показник становив більше 50%. Цього року можна чекати атак потужністю до 600 Гбіт/с, для чого теж буде

КПЗ_2025

задіяний метод посилення. Він же попереджає про перехід зловмисників на більш ефективні методи DDoS, які не вимагають використання великого числа зомбі-машин.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи протидії DDoS-атакам у комп’ютерній мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КбПЗ_2025

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

У даному розділі ми будемо розглядати міжмережні екрани, представлені в таблиці 2.1. У цій таблиці зазначена назва міжмережного екрана і його клас. Дана таблиця буде особливо корисна при підборі програмного забезпечення для захисту персональних даних.

Таблиця 2.1 – Список міжмережних екранів

Програмний продукт	Клас МЕ
МЕ « Блокпост-Екран»	4
Спеціальне програмне забезпечення міжмережний екран « Z-2», версія 2	2
Засіб захисту інформації TrustAccess	2
Засіб захисту інформації TrustAccess-S	2
Міжмережний екран StoneGate Firewall	2
Засіб захисту інформації Security Studio Endpoint Protection Personal Firewall	4
Програмний комплекс «Сервер безпеки CSP VPN Server.Версія 3.1»	3
Програмний комплекс «Шлюз безпеки CSP VPN Gate.Версія 3.1»	3
Програмний комплекс «Клієнт безпеки CSP VPN Client. Версія 3.1»	3
Програмний комплекс міжмережний екран «Ideco ICS 3»	4
Програмний комплекс «Трафік Інспектор 3.0»	3
Засіб криптографічного захисту інформації « Континент-ап». Версія 3.7	3
Міжмережний екран «Кіберсейф: Міжмережний екран»	3
Програмний комплекс « Інтернет-Шлюз Ideco ICS 6»	3
VipNet Office Firewall	4

КОПІЯ_2025

Для забезпечення 3 рівні захищеності ПД підійдуть міжмережні екрани не нижче 3 класи (або 4 класи, у випадку актуальності погроз 3-го типи й відсутності взаємодії ІС із Інтернетом). А для забезпечення 4 рівні захищеності підійдуть самі простенькі міжмережні екрани – не нижче 5 класу. По суті, кожний із представлених у таблиці 2.1 міжмережних екранів може використовуватися для забезпечення 1-3 рівнів захищеності за умови відсутності погроз 3-го типи й відсутності взаємодії з Інтернетом. Якщо ж є з'єднання з Інтернетом, то потрібний міжмережний екран як мінімум 3 класи.

Порівняння міжмережних екранів

Міжмережним екранам властивий певний набір функцій. От і подивимося, які функції надає (або не надає) той або інший міжмережний екран. Основна функція будь-якого міжмережного екрана – це фільтрація пакетів на підставі певного набору правил. Не дивно, але цю функцію підтримують всі брандмауери.

Також всі розглянуті брандмауери підтримують NAT. Але є досить специфічні (але від цього не менш корисні) функції, наприклад, маскування портів, регулювання навантаження, багатокористувальницьких режим роботи, контроль цілісності, розгортання програми в ActiveDirectory і вилучене адміністрування ззовні. Досить зручно, погодитеся, коли програма підтримує розгортання в ActiveDirectory – не потрібно вручну встановлювати її на кожному комп'ютері мережі. Також зручно, якщо міжмережний екран підтримує вилучене адміністрування ззовні – можна адмініструвати мережа, не виходячи з будинку, що буде актуально для адміністраторів, що звикли виконувати свої функції віддалено.

Напевно, читач буде здивований, але розгортання в ActiveDirectory не підтримують багато міжмережних екранів, представлених у таблиці 1, т же саме можна сказати й про інші функції, таких як регулювання навантаження й маскування портів.

КОПІЯ 2025

Адміністраторові часто однаково, якими додатковими функціями буде володіти міжмережний екран. Йому важливі наступні фактори:

1. Час захисту. Тут зрозуміло, чим швидше, тим краще.
2. Зручність використання. Не всі міжмережні екрани однаково зручні, що й буде показано в огляді.
3. Вартість. Часто фінансова сторона є вирішальною.
4. Строк поставки. Нерідко строк поставки залишає бажати кращого, а захистити дані потрібно вже зараз.

Безпека у всіх міжмережних екранів приблизно однакова, інакше в них би не було сертифіката.

Брандмауери в огляді

Далі ми будемо порівнювати три міжмережних екрани – VipNet Office Firewall, Кіберсейф Міжмережний екран і TrustAccess.

Брандмауер TrustAccess – це розподілений міжмережний екран із централізованим керуванням, призначений для захисту серверів і робочих станцій від несанкціонованого доступу, розмежування мережного доступу до ИС підприємства.

Кіберсейф Міжмережний екран – потужний міжмережний екран, розроблений для захисту комп'ютерних систем і локальної мережі від зовнішніх шкідливих впливів.

ViPNet Office Firewall 4.1 – програмний міжмережний екран, призначений для контролю й керування трафіком і перетворення трафіку (NAT) між сегментів локальних мереж при їхній взаємодії, а також при взаємодії вузлів локальних мереж з ресурсами мереж загального користування.

Час захисту

Що такий час захисту? По суті, цей час розгортання програми на всі комп'ютери мережі й час налаштування правил. Останнє залежить від зручності використання брандмауера, а от перше – від пристосованості його настановного пакета до централізованої установки.

КбПЗ_2025

Всі три міжмережних екрани поширюються у вигляді пакетів MSI, а це означає, що можна використовувати засоби розгортання ActiveDirectory для їхньої централізованої установки. Здавалося б все просто. Але на практиці виявляється, що немає.

На підприємстві, як правило, використовується централізоване керування міжмережними екранами. А це означає, що на якийсь комп'ютер установлюється сервер керування брандмауерами, а на інші встановлюються програми-клієнти або як їх ще називають агенти. Проблема вся в тому, що при установці агента потрібно задати певні параметри – як мінімум IP-адреса сервера керування, а може ще й пароль і т.д.

Отже, навіть якщо ви розгорнете MSI-файли на всі комп'ютери мережі, набудовувати їх однаково прийде вручну. А цього б не дуже хотілося, з огляду на, що мережа більша. Навіть якщо у вас усього 50 комп'ютерів ви тільки вдумайтеся – підійти до кожному ПК і настроїти його.

Як вирішити проблему? А проблему можна вирішити шляхом створення файлу трансформації (MST-файлу), він же файл відповідей, для MSI-файлу. От тільки ні VipNet Office Firewall, ні TrustAccess цього не вміють. Саме тому, до речі, у таблиці 2 зазначено, що немає підтримки розгортання Active Directory. Розгорнути те ці програми в домені можна, але потрібна ручна робота адміністратора.

Звичайно, адміністратор може використовувати редактори начебто Orca для створення MST-файлу.

КбПЗ-2025

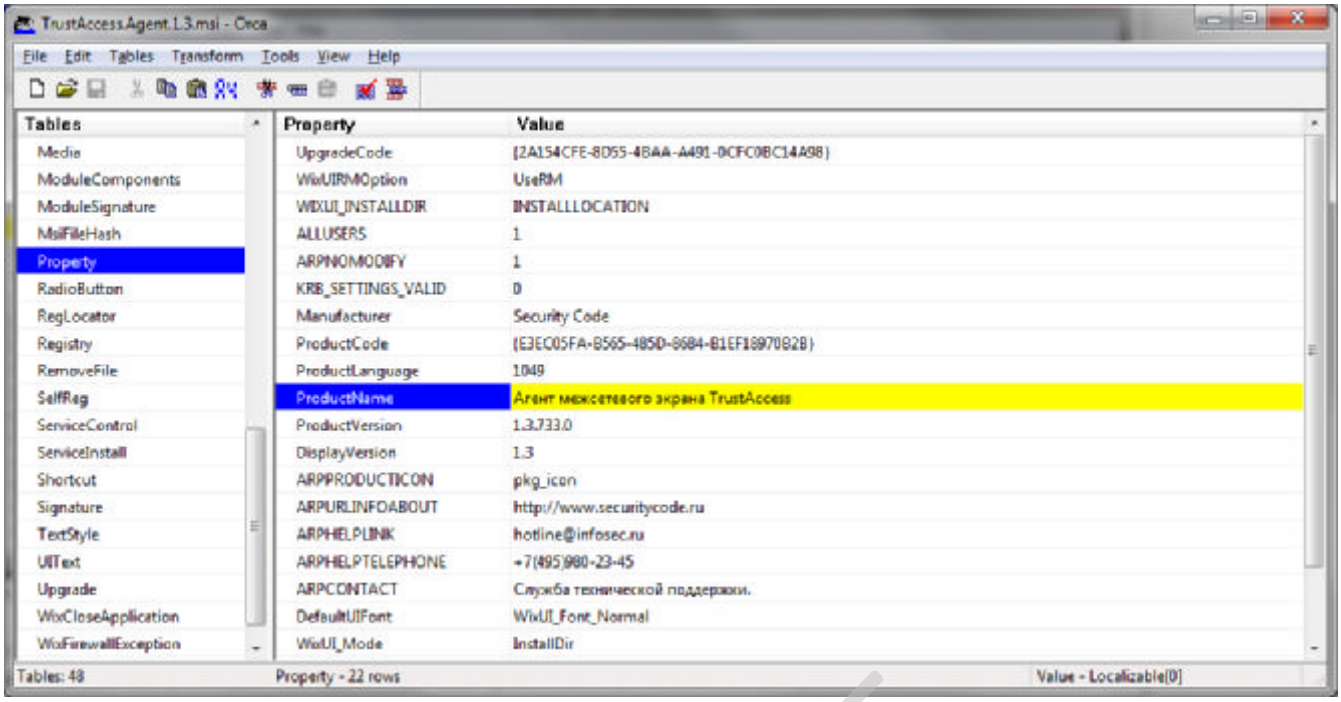

Рисунок 2.1 – Редактор Orca. Спроба створити MST-файл для TrustAccess.Agent.1.3.msi

Але невже ви думаєте, що всі так просто? Відкрив MSI-файл в Orca, підправив пари параметрів і одержав готовий файл відповідей? Не отут те було! По-перше, сам Orca просто так не встановлюється. Потрібно скачати Windows Installer SDK, з нього за допомогою 7-Zip витягти orca.msi і встановити його. Ви про це знали? Якщо ні, тоді вважайте, що витратили мінут 15 на пошук потрібної інформації, завантаження ПЗ й установку редактора. Але на цьому всі мучення не закінчуються. В MSI-файлу безліч параметрів. Подивитеся на рисунок 2.1 – це тільки параметри групи Property. Який з них змінити, щоб указати IP-адреса сервера? Ви знаєте? Якщо ні, тоді у вас два варіанти: або вручну настроїти кожний комп'ютер або звернутися до розроблювача, чекати відповідь і т.д. З огляду на, що розроблювачі іноді відповідають досить довго, реальний час розгортання програми залежить тільки від швидкості вашого переміщення між комп'ютерами. Добре, якщо ви завчасно встановили інструмент вилученого керування – тоді розгортання пройде швидше.

КОПІЯ 2025

Кіберсейф Міжмережний екран самостійно створює MST-файл, потрібно лише встановити його на один комп'ютер, одержати заповітний MST-файл і вказати його в груповій політиці. Про те, як це зробити, можна прочитати в статті «Розмежування інформаційних систем при захисті персональних даних». За якісь півгодини (а те й менше) ви зможете розгорнути міжмережний екран на всі комп'ютери мережі.

Саме тому Кіберсейф Міжмережний екран одержує оцінку 5, а його конкуренти – 3 (спасибі хоч інсталятори виконані у форматі MSI, а не .exe).

Зручність використання

Брандмауер – це не текстовий процесор. Це досить специфічний програмний продукт, використання якого зводиться до принципу «установив, настроїв, забув». З одного боку, зручність використання – другорядний фактор. Наприклад, iptables в Linux не можна назвати зручним, але адже ним же користуються? З іншого боку – чим зручніше брандмауер, тим швидше вийде захистити і виконувати деякі функції по її адмініструванню.

Що ж, давайте подивимося, наскільки зручні розглянуті міжмережні екрани в процесі створення й захисту.

Почнемо ми з VipNet Office Firewall, що, на наш погляд, не дуже зручний. Виділити комп'ютери в групи можна тільки по IP-адресах (рисунок 2.2). Інакше кажучи, є прив'язка до IP-адрес і вам потрібно або виділяти різні у різні підмережі, або ж розбивати одну підмережу на діапазони IP-адрес. Наприклад, є три : Керування, Бухгалтерія, IT. Вам потрібно настроїти DHCP-сервер так, щоб комп'ютерам із групи Керування «лунами» IP-адреси з діапазону 192.168.1.10 – 192.168.1.20, Бухгалтерія 192.168.1.21 – 192.168.1.31 і т.д. Це не дуже зручно. Саме за це з VipNet Office Firewall буде знятий один бал.

КБПЗ_2025

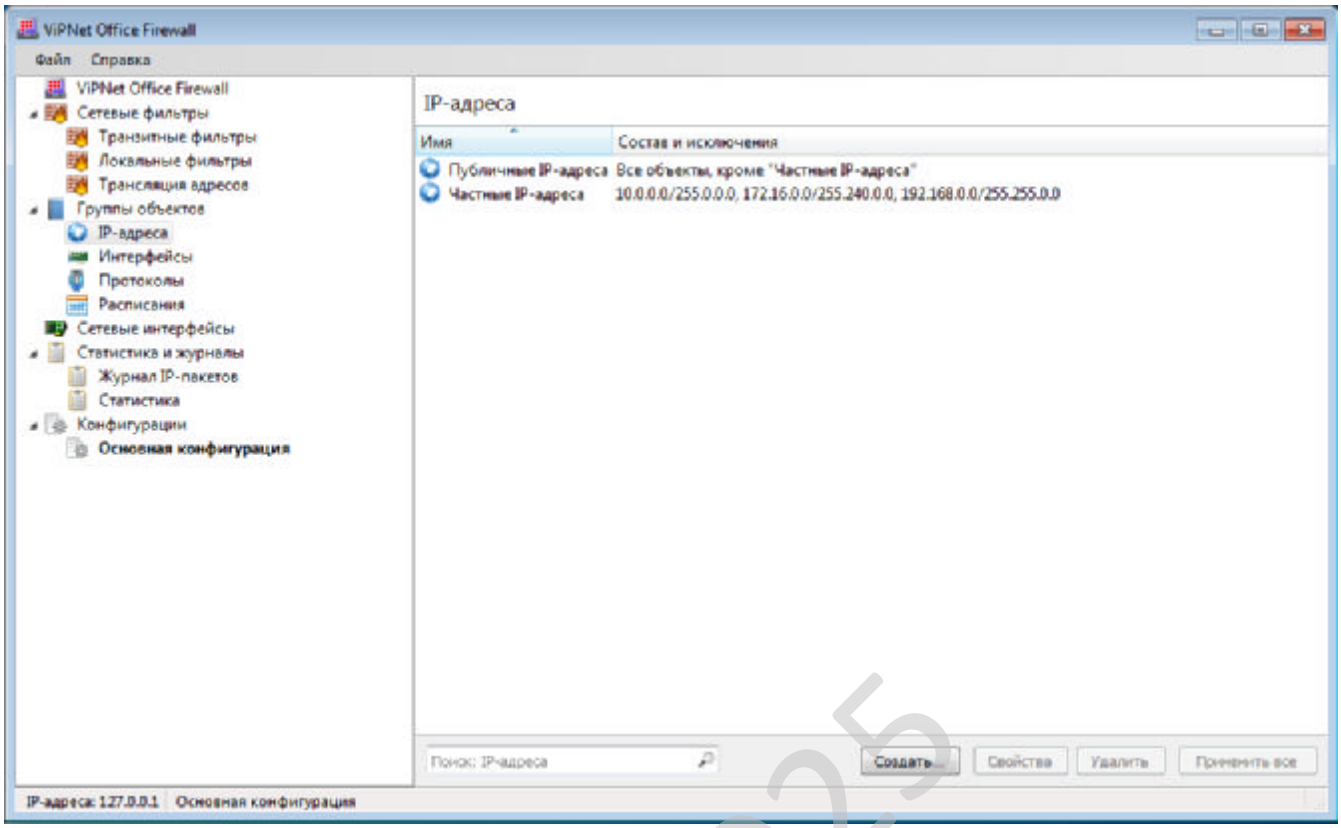

Рисунок 2.2 – При створенні груп комп'ютерів спостерігається явна прив'язка до IP-адреси

У міжмережному екрані Кіберсейф, навпаки, немає ніякої прив'язки до IP-адреси. Комп'ютери, що входять до складу групи, можуть перебувати в різних підмережах, у різних діапазонах однієї підмережі й навіть перебувати за межами мережі. Подивитеся на рисунок 2.3. Філії компанії розташовані в різних містах. Створити групи дуже просто – досить перетягнути імена комп'ютерів у потрібну групу й нажати кнопку **Застосувати**. Після цього можна нажати кнопку **Встановити правила** для формування специфічних для кожної групи правил.

КБПЗ_2025

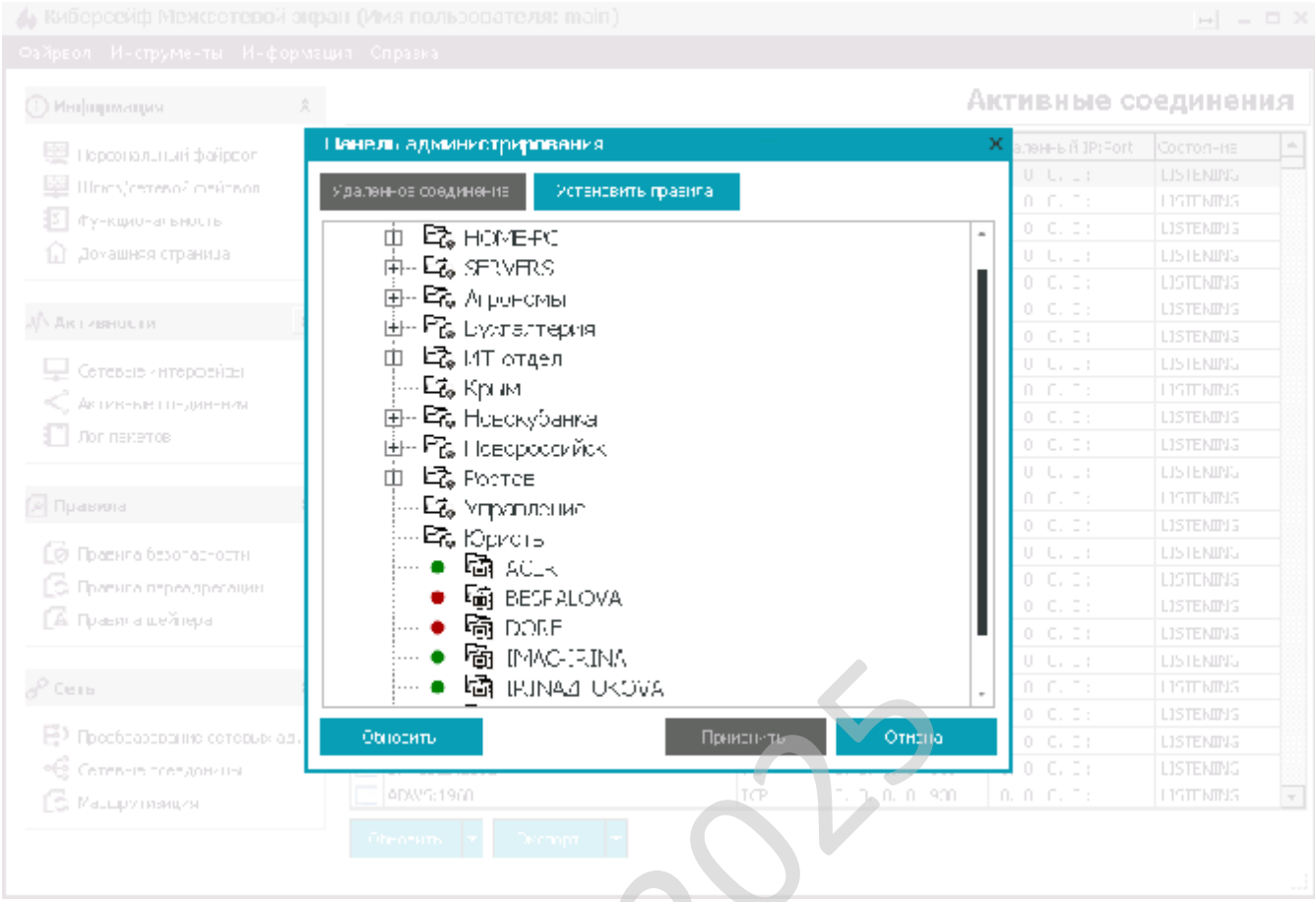

Рисунок 2.3 – Керування групами в Кіберсейф Міжмережний екран

Що стосується TrustAccess, те потрібно відзначити тісну інтеграцію із самою системою. У конфігурацію брандмауера імпортуються вже створені системні групи користувачів і комп'ютерів, що полегшує керування міжмережним екраном у середовищі ActiveDirectory. Ви можете не створювати у самому брандмауері, а використовувати вже наявні групи комп'ютерів у домені Active Directory.

КПЗ 2025

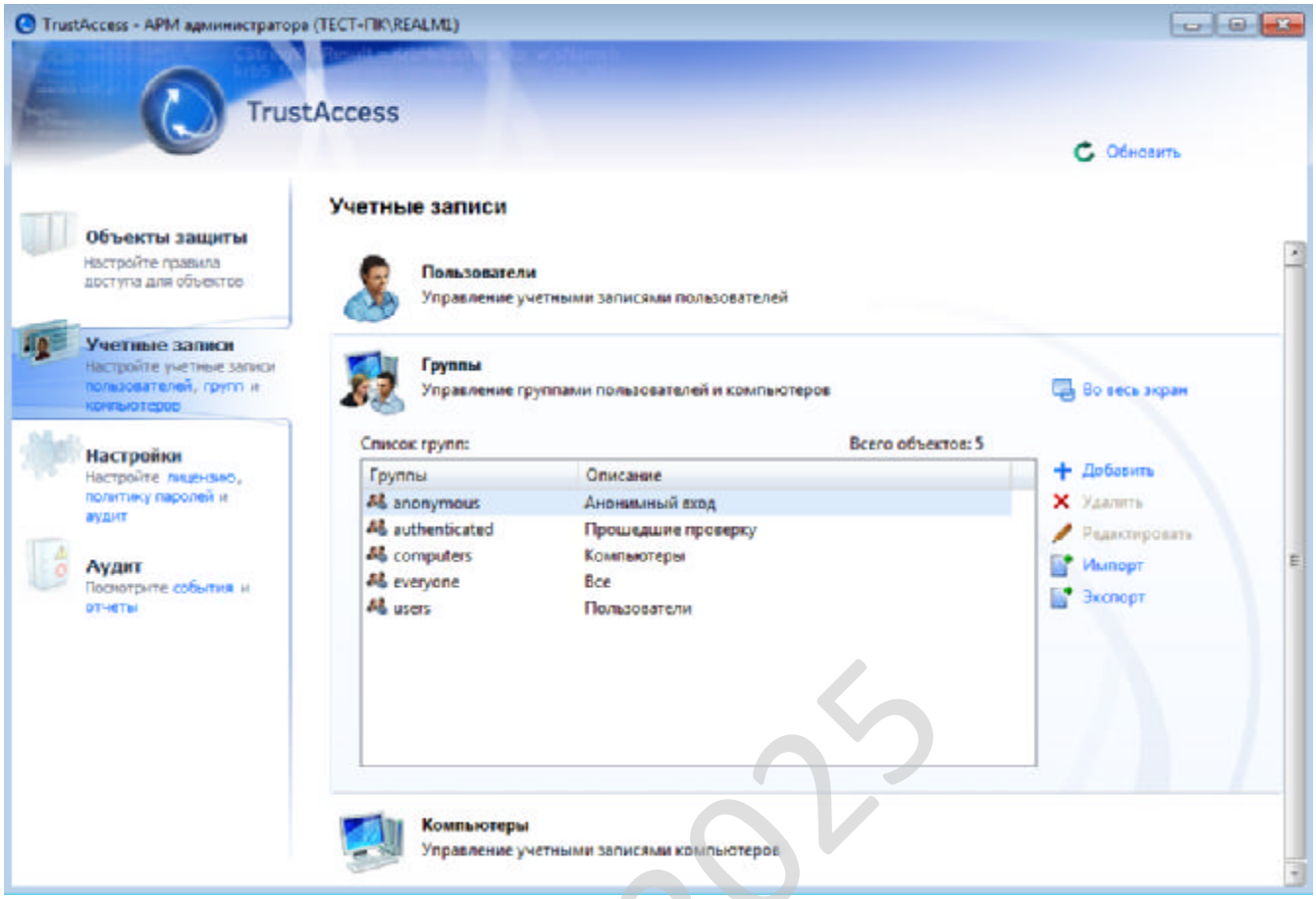

Рисунок 2.4 – Групи користувачів і комп'ютерів (TrustAccess)

Всі три брандмауери дозволяють створювати так звані розклади, завдяки яким адміністратор може настроїти проходження пакетів за розкладом, наприклад, заборонити доступ до Інтернету в неробочий час. В VipNet Office Firewall розклади створюються в розділі Розкладу (рисунок 2.5), а в Кіберсейф Міжмережний екран час роботи правила задається при визначенні самого правила (рисунок 2.6).

КПЗ_2025

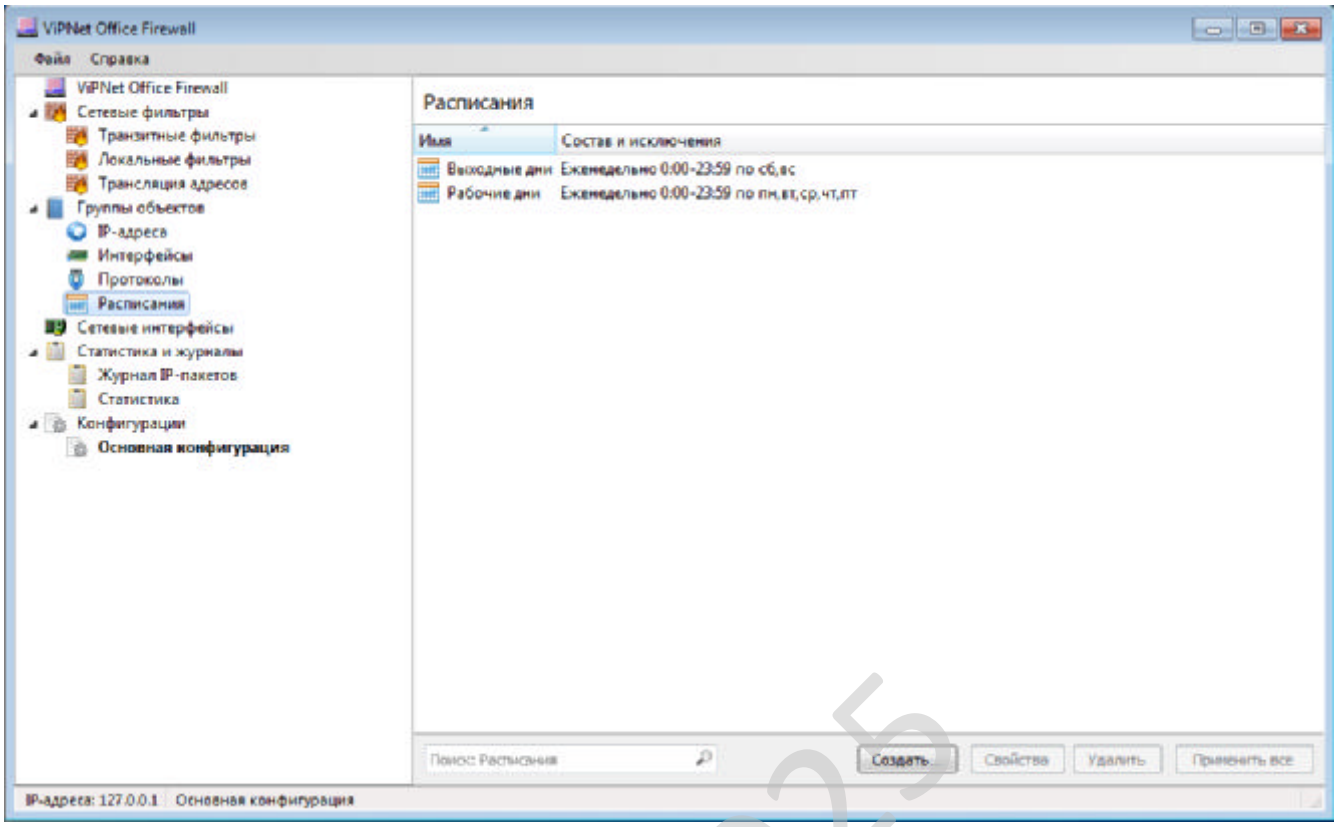

Рисунок 2.5 – Розклад в VipNet Office Firewall

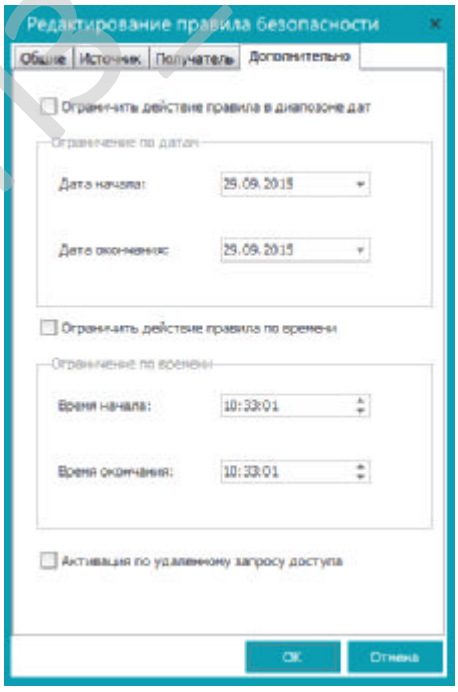

Рисунок 2.6 – Час роботи правила в Кіберсейф Міжмережний екран

КБПЗ 2025

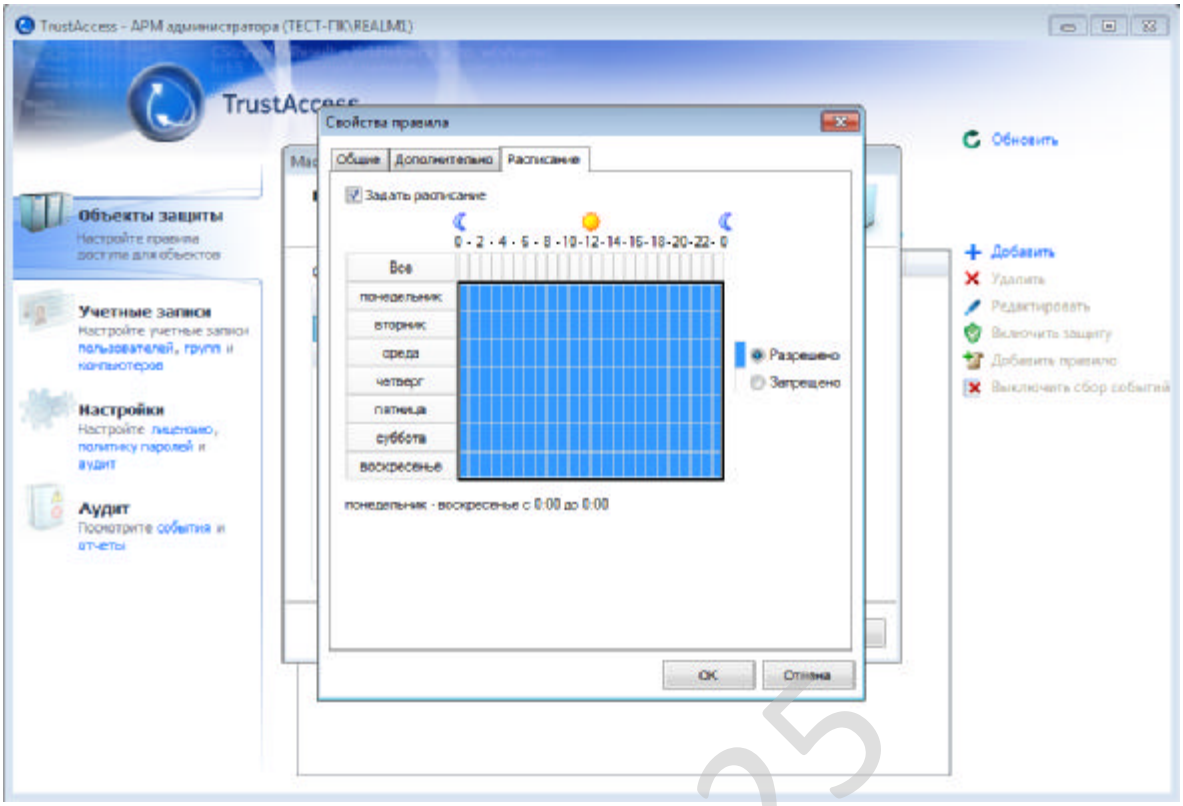

Рисунок 2.7 – Розклад в TrustAccess

Всі три брандмауери надають дуже зручні засоби для створення самих правил. А TrustAccess ще й надає зручний майстер створення правила.

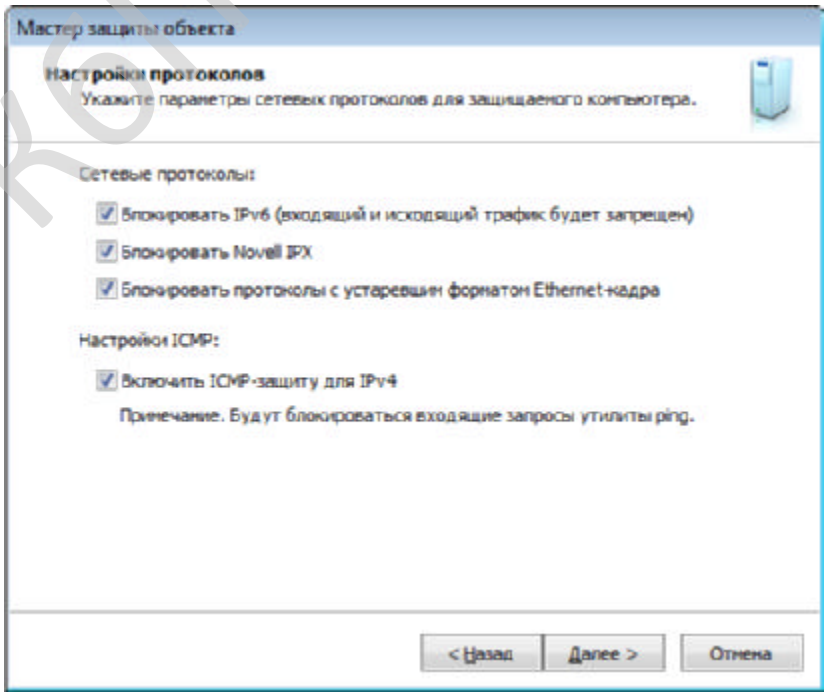

Рисунок 2.8 – Створення правила в TrustAccess

КОПЗ 2025

Глянемо на ще одну особливість – інструменти для одержання звітів (журналів, балок). В TrustAccess для збору звітів і інформації про події потрібно встановити сервер подій (EventServer) і сервер звітів (ReportServer). Не те, що це недолік, а скоріше особливість («feature», як говорив Білл Гейтс) даного брандмауера. Що стосується, міжмережних екранів Кіберсейф і VipNet Office, те обидва брандмауери надають зручні кошти перегляду журналу IP-пакетів. Різниця лише в тому, що в Кіберсейф Міжмережний екран спочатку відображаються всі пакети, і ви можете отфильтровать потрібні, використовуючи можливості убудованого в заголовок таблиці фільтра (рисунок 2.9). А в VipNet Office Firewall спочатку потрібно встановити фільтри, а потім уже переглянути результат.

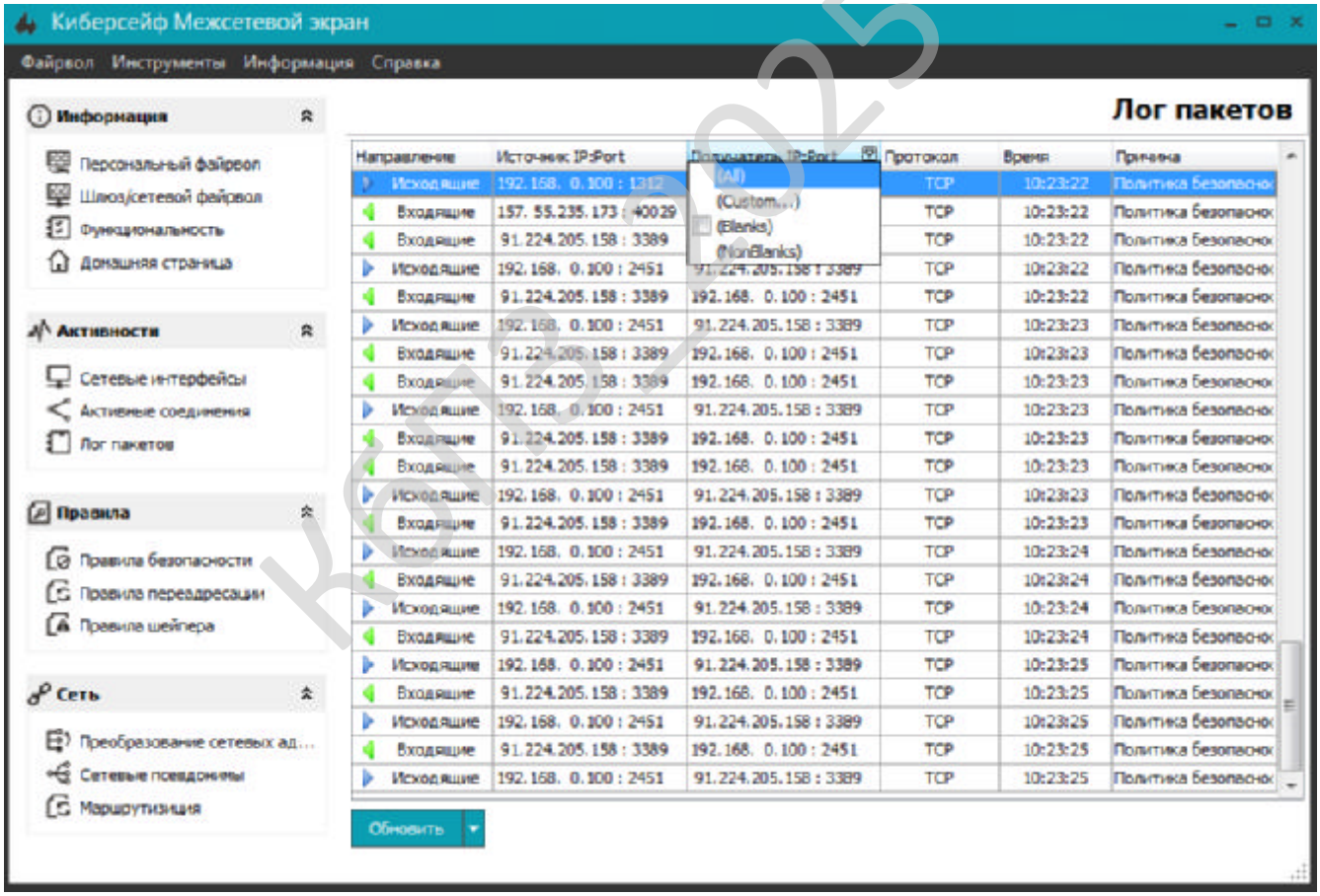

Рисунок 2.9 – Керування журналом IP-пакетів у Кіберсейф Міжмережний екран

КРБ-2025

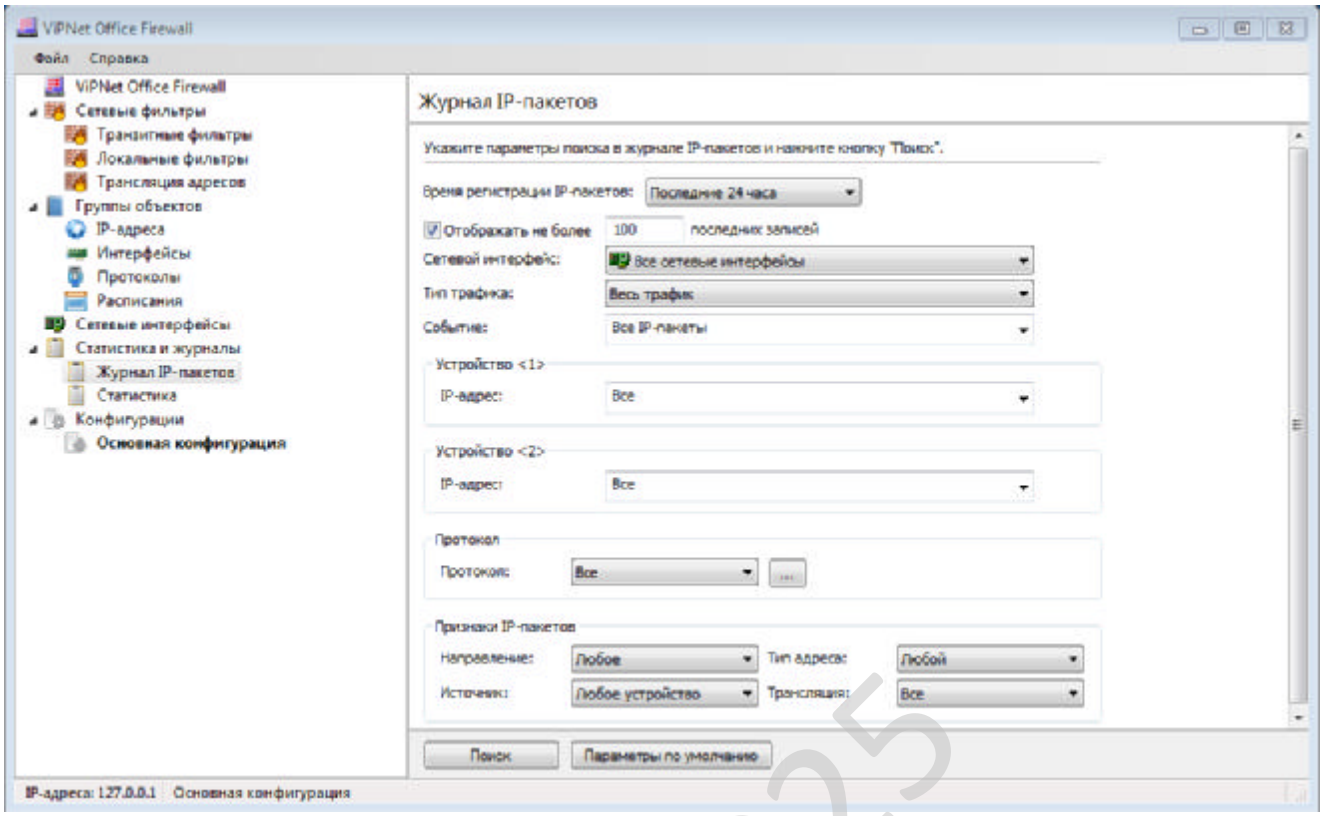

Рисунок 2.10. Керування журналом IP-пакетів в VipNet Office Firewall

З міжмережного екрана Кіберсейф довелося зняти 0.5 бала за відсутність функції експорту журналу в Excel або HTML. Функція далеко не критична, але іноді корисно просто й швидко експортувати з журналу кілька рядків, наприклад, для «розбору польотів».

Якщо орієнтуватися тільки за вартістю продукту й технічної підтримки, то вибір очевидний – Кіберсейф Міжмережний екран. Кіберсейф Міжмережний екран має оптимальне співвідношення функціонал/ціна. З іншого боку, якщо вам потрібна підтримка Secret Net, те потрібно дивитися убік TrustAccess. А от VipNet Office Firewall можемо порекомендувати хіба що як гарного персонального брандмауера, але для цих цілей існує безліч інших і до того ж безкоштовних рішень.

КОПЗ – 2025

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктноорієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктноорієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веброзробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв’язанні проблеми. Їм не

КбПЗ-2025

потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

КПЗ - 2025

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і

КПЗ – 2025

заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов'язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати,

КОПЗ-2025

перетворювати, розділяти, об’єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веброзробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

КБПЗ-2025

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи протидії DDoS-атакам у комп’ютерній мережі.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

КОПЗ - 2025

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Там, де використовуються автоматизовані засоби атаки, завжди можуть бути розроблені й автоматизовані засоби захисту. Зокрема, деякі виробники випускають спеціальні пристрої, здатні блокувати непродуктивні запити. Такі пристрої є в арсеналі компаній Cisco, Arbor Networks, CloudShield і інших вендорів. Подібні рішення здійснюють фільтрацію паразитного трафіку на високих швидкостях і призначені в основному для провайдерів – вони повинні встановлюватися не перед корпоративним сайтом, а якнайближче до джерела непродуктивних запитів. Зокрема, захист за допомогою подібних пристроїв може запропонувати оператор, що підключає сайт до Інтернету. Тому хостинг варто замовляти в того провайдеру, що забезпечує захист від DDoS-атак.

Компанії можуть скористатися послугами спеціальних сервісів по захисту від DDoS, що попередньо очищають трафік від сторонніх запитів. Системи захисту сервісу контролюють запити, що надходять на сайт клієнта, і намагаються відсіяти ті з них, що дуже схожі на сторонні. Це дозволяє фільтрувати трафік не перед сайтом клієнта, а на далеких підступах до нього. Така розподілена мережа фільтрації допомагає захиститися від найбільш потужних розподілених атак. Хостинг-провайдер не в змозі забезпечити подібний рівень фільтрації – у випадку дуже сильної атаки хостери найчастіше самі виключають сайт, на який виробляється атака, щоб зберегти доступність інших.

Втім, не варто забувати й про протидію саботажу. Хоча такі атаки тільки почали з'являтися, про захист від них уже варто подумати. Рекомендація проста – потрібна надійна автентифікація адміністраторів хмарних сервісів і керованих через Web засобів захисту. Дуже важливо зробити процедуру автентифікації користувача більше захищеною. Один з варіантів – використовувати для цього

КПЗ – 2025

спеціальні персональні пристрої, усередині яких зберігається ідентифікатор користувача. У цьому випадку, на відміну від зв'язування логін-пароль, щоб скористатися обліковим записом, потрібно мати сам пристрій і знати пароль до нього. Важливим моментом є неможливість клонувати такий пристрій з ідентифікатором користувача усередині. Не варто забувати й про захист пошти, на яку реєструються облікові записи користувачів хмарних сервісів. Не рекомендується для цього використовувати Web-пошту – краще спеціально створити невеликий власний поштовий домен і на ньому настроїти більше складні механізми автентифікації.

Треба пам'ятати й про юридичний DoS-атаці, що може протікати по такому сценарії: після злому CMS створюється додатковий сайт, що наповнюється забороненою інформацією, про наявність якої повідомляється. У результаті IP-адреса сайту попадає в чорний список, через що в користувачів виникають проблеми з доступом, у тому числі й до основного сайту. Для захисту від такого сценарію потрібно контролювати CMS – як мінімум мати власну встановлену систему, а не орендовану на спільно використовуваному сервері. При розміщенні CMS у хмарі варто передбачити механізми контролю конфігурації й пошуку стороннього коду.

У цілому ж варто пам'ятати, що DoS-атака – це не тільки цілеспрямоване переповнення каналів, але й результат неправильного керування обчислювальними ресурсами сайту, тому не слід допускати появи в його архітектурі вузьких місць.

КОПЗ_2025

3.2 Розробка структурної схеми

У процесі пошуку підходящого засобу захисту від атак DDoS лише деякі роблять вибір на користь апаратних рішень, адже переваги захисту із хмари очевидні. Приміром, відпадає необхідність у трудомісткому переміщенні апаратного забезпечення – хмарні рішення не мають потреби в установці, налаштуванні й обслуговуванні встаткування. Загалом, на відміну від апаратних

систем захисту від атак DDoS, стають зайвими всі інвестиції в надлишково оснащені фізичні системи захисту, а також зусилля по їхньому адмініструванню, технічному обслуговуванню, експлуатації й змісту. Ще один аргумент на користь захисту із хмари – безпроблемне масштабування: при необхідності нарощування відповідних ресурсів забезпечується сервісом-провайдером. Клієнтові не доводиться витрачатися на нове апаратне забезпечення або вкладати кошти в придбання додаткових обчислювальних потужностей.

В оптимальному випадку надання послуг із захисту від атак DDoS із хмари здійснюється швидко й без проблем, оскільки технічні зміни зачіпають тільки параметри DNS і маршрутизації. При бажанні можна з такою ж легкістю відмовитися від цього захисту.

У цей час хмарні послуги із захисту від атак DDoS уже пропонують чимало компаній, але лише деякі з них забезпечують досить ефективний захист, що дозволяє в повному обсязі убезпечити Web-сторінки від всіх можливих видів атак (див. Рисунок 3.1). Більшість постачальників не надають ніяких точних відомостей про принцип роботи своїх систем, а виходить, і про якість забезпечуваної ними захисту. При цьому знайомство з деталями було б дуже корисно при виборі найбільш підходящого рішення, а кожному IT-фахівцеві варто з'ясувати, чи відповідає пропоноване рішення для захисту від атак DDoS наступним умовам.

Точні алгоритми фільтрації. Використовувані алгоритми фільтрації є вирішальним критерієм для вибору гарного захисту від атак DDoS. Важливо, щоб фільтри блокували не весь трафік до сторінки, а лише шкідливий трафік DDoS. При цьому засобу динамічного аналізу поводження IP-трафіку забезпечують більше висока якість захисту, чим статичні механізми фільтрації. Щоб імовірність відмови зберігалася на мінімальному рівні, необхідні додаткові індивідуальні налаштування. Це дозволяє краще контролювати мережний трафік, а також підрозділяти його на «бажаний», «небажаний», «володіючий особливими повноваженнями» і т.д. Використовувані механізми фільтрації повинні бути максимально зрозумілі і наочними, щоб можна було перевірити, чи не відсіває система захисту від атак DDoS і добропорядних відвідувачів.

КПІ_2025

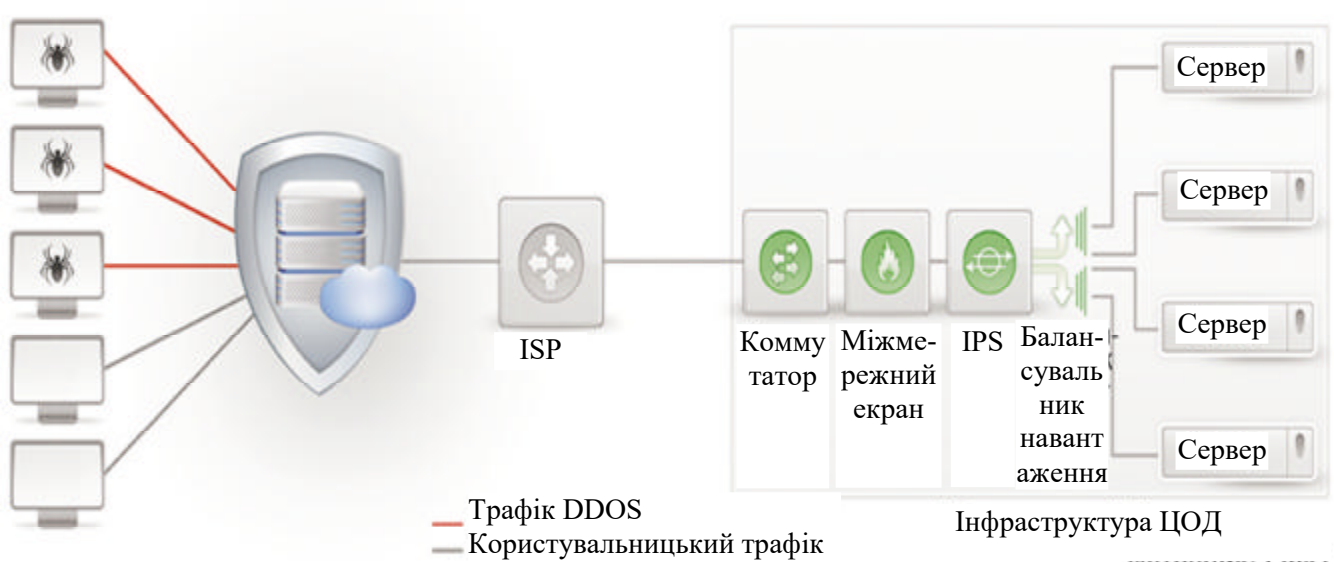

Рисунок 3.1 – Структурна схема системи

Висока пропускна здатність. Зловмисники іноді задіють десятки тисяч інфікованих комп'ютерів, тому потужність атак може досягати 5-10 Гбіт/с. Через їхню величезну кількість ідентифікація окремих атакуючих комп'ютерів неможлива, та й не має змісту. Найчастіше зараження комп'ютера трояном або ботом відбувається за місяці до початку атаки. Як тільки керуючий атакою зловмисник (Master) вибирає жертву й подає сигнал до нападу, всі комп'ютери, заражені шкідливою програмою, починають діяти.

У своїй сукупності ці комп'ютери становлять величезну армію з колосальною ударною міццю, здатної блокувати всю діяльність сайту, що атакується. Тому важливо, щоб постачальник системи захисту забезпечував достатню пропускну здатність, адже як масштаб нападу, так і ресурси захисної системи можна зрівняти з м'язовою силою людини: щоб протистояти об'єднаній атаці десятків тисяч комп'ютерів, захисний кластер повинен мати чималий фізичний ресурс.

Графічний користувальницький інтерфейс. Гарна система захисту повинна мати інтерфейс для моніторингу відбиття атаки в реальному часі. Тоді користувачі зможуть контролювати діяльність засобів захисту, відслідковувати види атак, їхня тривалість, характер протікання, інтенсивність, регіон

КПЗ-2025

походження, а при необхідності вручну встановлювати заборони або привласнювати особливих прав для окремих IP-адрес

Крім того, сервіс-провайдер повинен щомісяця надавати звіт про всю активність DDoS, де відбивався б характер подій і їхній обсяг, – це дозволить контролювати ситуацію й здійснювати подальше стратегічне планування захисту від подібних атак. Тільки такий звіт дає можливість відстежити, чи не блокувала система захисту доступ добропорядним користувачам.

Масштабованість. Обсяг атак може значно варіюватися, оскільки зловмисники використовують для нападу на Web-сторінки інструменти, що діють із різним ступенем ефективності. Принаймні, такі виводи можна зробити на підставі досвіду численних провайдерів засобів захисту від атак DDoS. Пропускна здатність захисних систем повинна масштабуватися від одного до декількох десятків гігабіт у секунду, тобто підтримувати можливість швидкого розширення, щоб при необхідності впоратися з великомасштабними атаками. Таким чином, необхідно, щоб фільтруючий кластер сервіс-провайдеру підтримував досить високий ступінь масштабування, що дозволяє в серйозній ситуації впоратися з атаками, потужність яких перевищує 100 Гбіт/с. Якщо фільтруючий кластер занадто малий, то у випадку масованої атаки постраждають і ті, на кого атака не націлювалася. Реалізація моделі, коли додаткові ресурси підключаються лише в самих крайніх випадках, дозволяє обійтися порівняно невеликими витратами на підтримку необхідної пропускної здатності систем захисту від атак DDoS, при цьому забезпечується можливість залучення додаткових потужностей в екстрених ситуаціях.

Сценарії атак DDoS постійно видозмінюються. Зловмисники застосовують усе більше витончені методи, змушуючи постачальників засобів захисту вирішувати нові й постійно, що ускладнюються завдання. Прикладом можуть служити так звані повільні атаки Slow-DDoS. Наприклад, атака Slowloris націлена на те, щоб зберігати з'єднання сервера, що атакується, відкритими. Зловмисник досягає цього шляхом відправлення незавершених запитів HTTP, і поступово на

КОПЗ-2025

сервер надходить усе більше й більше таких запитів. Кількість відкритих з'єднань, підтримуваних сервером, обмежено, він змушений відхиляти правомірні запити від Web-Браузерів, його доступні ресурси виснажуються, і він перестає виконувати свої функції. Але по формальних характеристиках трафік даних відповідає вимогам протоколів і швидкість запитів теж не перевищує дозволених значень, тому традиційні системи захисту від атак DDoS не здатні розпізнати атаки такого типу.

Успіх досягається тільки при використанні новітніх методів захисту: у ході проміжного збереження здійснюється перевірка заголовків (Header) і тіла (Body) запитів, на підставі аналізу яких відразу ж закриваються недійсні з'єднання. Крім того, оптимізація розмірів і продуктивності окремих з'єднань дозволяє підтримувати набагато більше їхнє число одночасно. Від подальших атак DDoS Low and Slow захистять безперервний аналіз трафіку і його порівняння з режимом нормальної роботи Web-сторінки, тому що «повільні» атаки DDoS можна розпізнати на основі незвичайного поводження користувачів і потім перешкодити їм.

Готові предналаштовані рішення для захисту від атак DDoS недостатньо гнучкі, щоб підлаштовуватися під еволюцію методів нападу. Обране рішення повинне підтримувати можливість додавання нових сценаріїв атак, що дозволить уникнути неприємних сюрпризів. Даремно використовувати застарілі засоби захисту, як і антивірусну програму, що останній раз обновлялася кілька років назад. Тому для забезпечення надійного захисту потрібний кваліфікований персонал, здатний відразу ж реагувати на нові форми атак DDoS. У деяких постачальників захисних систем є власні експертні центри із групами екстреного реагування на нові види погроз.

Ідея переносу захисту від DDoS у хмари, безсумнівно, розбурхує розуми теоретиків в області інформаційних технологій, але об'єктивна реальність змушує спуститися з позахмарних висот на землю. Коли-небудь ми дійсно виявимося у світі програмно керованих мереж і всі наші IT-ресурси будуть формуватися на

КОПЗ - 2025

основі надійних розподілених хмарних технологій. Але поки реалії такі: будь-яка хмара – це набір обчислювальних ресурсів, розташованих в одному або декількох центрах обробки даних, найчастіше не дуже добре зв'язаних між собою. Понадіявшись на засоби захисту від DDoS, розміщені в такій хмарі, можна виявитися в такій ситуації, коли вони виявляться абсолютно марні, оскільки вся ємність каналів, що зв'язують ЦОД із зовнішнім миром, буде зайнята паразитним трафіком, через що всі його ресурси виявляться недоступні ззовні.

Атаки DoS/DDoS стають усе могутнішими, причому зловмисники використовують при їхньому здійсненні множинні крапки входу в мережу, вектори й типи атак. Це робить атаки особливо небезпечними, а їхнє відбиття стає складним і кропітким завданням. Організація захисту власними силами для будь-якої компанії, не пов'язаної із цим завданням професійно, означає значні витрати на встаткування, ПЗ й експертизу, які, швидше за все, так і не приведуть до бажаного результату.

Незважаючи на різноманіття сценаріїв DDoS, сучасні засоби захисту мережі дозволяють запобігти найпоширенішим типам таких атак. Традиційні IPS і міжмережні екрани мають обмежені функціональні можливості для блокування DDoS – із захистом ефективніше всього справляються спеціалізовані пристрої й системи глобального оповіщення про джерела й типи атак. Вони здатні в режимі реального часу аналізувати величезний потік даних і фільтрувати паразитні пакети визначати й блокувати трафік, що є результатом роботи сценаріїв або роботів; вишиковувати ешелоновану систему захисту, що враховує весь спектр можливих алгоритмів атаки.

Захист від атак DoS/DDoS повинен бути комплексним й у кожному випадку вимагає детального аналізу. Якщо оператор центра обробки даних не має достатній досвід і спеціальні технічні засоби для боротьби із цим явищем, DDoS-атака може викликати загальну неприступність мережних пристроїв і, як наслідок, IT-інфраструктури його клієнтів. ЦОД всі частіше зіштовхуються із ситуацією, коли потужність атаки досягає десятків гігабіт у секунду. У таких

КБПЗ-2025

умовах навіть спеціальне дороге устаткування, що перебуває в центрі обробки даних, не в змозі впоратися із проблемою – потрібні ідентифікація паразитного трафіку й наступна його фільтрація на всіх територіально розподілених вузлах магістральних мереж. Характерно, що необхідність захисту інформаційних ресурсів від DDoS сьогодні стає проблемою національного масштабу.

Багато вітчизняних компаній здатні надати власні рішення й апаратно-програмні комплекси, які в стані забезпечити широкомасштабний захист IT-ресурсів від DDoS на рівні магістральних мереж. Довіривши рішення цієї проблеми надійному операторові, клієнти могли б дістати ефективні кошти захисту. Однак для оснащення магістральних вузлів провідних операторів мереж зв'язку й крапок обміну трафіком високопродуктивним вітчизняним устаткуванням і програмним забезпеченням для ідентифікації й очищення трафіку необхідна організаційно-правова й фінансова підтримка держави.

Наявність ефективного захисту від атак DDoS особливо необхідно тим підприємствам, діяльність яких прямо пов'язана з підключенням до мережі Інтернет. Але й проста доступність сайту компанії теж має вкрай важливе значення для підтримки ділової репутації. Потенційних зловмисників безліч, і для деяких з них незахищена Web-сторінка є ласим шматочком. Ініціаторами нападу можуть бути хакери, здирники або – що зустрічається досить часто – підлітки, що бажають спробувати свої сили. При виборі постачальника, що пропонує дійсно ефективний захист від атак DDoS, варто зібрати як можна більше докладну інформацію, зіставити з вищевказаними критеріями й ретельно їх перевірити ще раз.

КПЗ-2025

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Так, як функціональна схема є більш подобним описом функціональних можливостей структурної схеми, то вона буде представляти собою, більш детальний варіант

структурної схеми.

З рисунку видно, що розроблена система складається з наступних частин:

– Блок моніторингу мережі.

– Блок аналізу мережної статистики.

– Блок визначення топології мережі.

– Блок виявлення аномального поводження трафіку.

– Блок визначення виду атаки.

– Блок зберігання результатів.

Блок аналізу мережної статистики

Блок збирання наступної інформації:

– Основна статистика (Summary).

– Ієрархія протоколу (Protocol Hierachy).

– Сеанси обміну пакетами (Conversations).

– Точки призначення (Endpoints).

– Графіки I/O (IO Graphs).

– Список сеансів обміну пакетами (Conversation List).

– Список точок призначення (Endpoint List).

– Час чекання відповіді від сервісу (Service Response Time).

– RTP.

– SIP.

– Виклики VoIP (VoIP Calls).

– Призначення (Destination).

– Графік потоку (Flow Graph).

– HTTP.

– IP-адреса (IP address).

– Довжина пакету (Packet Length).

– Тип порту (Port Type).

Розпишемо їх більш детально.

1. Основна статистика. Доступні такі елементи основної статистики, як:

КБПЗ-2025

– Властивості захоплених файлів.

– Час захвату.

– Інформація про фільтр захвату.

– Інформація про фільтр відображення.

2. Ієрархія протоколу. Статистика ієрархії протоколу допомагає аналізувати пакети, розбиваючи відображені дані, які належать чинному рівню OSI.

3. Сеанси обміну пакетами. Якщо ви використовуєте протокол TCP/IP або програму, яка працює із цим протоколом, ви маєте побачити чотири активних вкладок для обміну пакетами за допомогою Ethernet, IP, TCP та UDP. «Діалог» між комп'ютерами відображає трафік між двома активними хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами, наприклад, «Ethernet:6».

4. Точки призначення. Точки призначення забезпечують статистику даними про відправку та прийом пакетів. Номер, зазначений на вкладці після назви протоколу, вказує на кількість точок призначення. Наприклад, «Ethernet:6».

5. Графіки I/O. Основний графік може бути отриманий за допомогою команди «IO graphs» (Графіки I/O). Ще декілька графіків можуть бути додані у тому ж вікні на основі фільтрів відображення.

6. Час чекання відповіді від сервісу. 13 протоколів доступні для глибокого аналізу.

7. RTP. RTP (Real-time Transport Protocol, протокол передачі у реальному часі, RFC 3550) – це протокол для передачі звука та відео через IP-мережу. Він працює у початку протоколу дейтаграм користувача (User Datagram Protocol, UDP). Він часто використовується у сукупності з протоколами SIP або H.233, забезпечуючи виконання сигнальних завдань.

8. SIP. SIP (Session Initiation Protocol, протокол встановлення сесії, RFC 3261) – це сигнальний протокол, який оголошує відео– або VoIP-сесії. Він працює

КБПЗ–2025

разом із протоколом RTP, який використовується для передачі мультимедійних даних.

9. Виклики VoIP.

VoIP (Voice over IP, голосовий зв'язок за допомогою Інтернету) взагалі використовує два типи протоколів:

– сигнальні протоколи, такі, як SIP або H.323

– переносні протоколи, наприклад, RTP

10 Призначення. Відображення усіх IP-адрес призначення мережевих пакетів.

11. Графік потоків. Графіки потоків забезпечує послідовний аналіз TCP-з'єднань. Перші три строки містять оголошення TCP– з'єднання з послідовностями «SYN», «SYN ACK» та «ACK».

12 HTTP. HTTP (Hypertext Transfer Protocol, протокол передачі гіпертексту) – це протокол типу «клієнт-сервер», який використовується для передачі HTML-файлів. HTTP-клієнт (у більшості випадків це web-браузер) відсилає HTTP-запит до web-серверу із полем «URL», який допомагає знайти потрібний файл. Web-сервер відповідає HTTP-пакетом та забезпечує клієнт необхідною web-сторінкою.

Меню «HTTP» містить три підменю:

– «Load Distribution» (Розподіл пакетів).

– «Packet Counter» (Лічильник пакетів).

– «Requests» (Запити).

14 IP-адреса. Відображення IP-адреси джерела або призначення мережевих пакетів.

15. Довжина пакету.

16. Тип порту. Відображення статистики портів TCP або UDP.

Блок визначення топології мережі

Блок визначення топології мережі:

КПЗ 2025

– Блок використання відомостей із загальної системи моніторингу мережі, а не опитування пристрою додатково.

– Блок складання списку пристроїв у мережі, автоматично, ґрунтуючись на дані системи моніторингу.

– Блок побудови топології мережі, за станом на задану дату й відстеження змін у топології протягом часу.

– Блок автоматичного визначення рівнів ієрархії пристроїв у мережі, з виділенням периферійних, проміжних і центральних вузлів;

– Блок побудови топології мережі, незалежно від використовуваної системи моніторингу й програмно-апаратних платформ;

– Блок комбінувати показників, на основі яких визначаються зв'язки між пристроями, і при їхньому обчисленні виконувати перевірку на значимість із використанням статистичних критеріїв.

Блок виявлення аномального поводження трафіку

Блок виявлення аномального поводження трафіку:

– Блок визначення профілю поводження нормального трафіку.

– Блок заміни направлення трафіку.

– Блок усунення аномального трафіку.

При первісному розгортанні рішення по DDoS адміністратор створює профіль поводження нормального трафіку. Цей процес іменується навчанням. Компанія використовує додатки звичайним образом протягом 24 годин протягом одного тижня, і трафік додатка проходить через Детектор аномалій трафіку. У період навчання Детектор аномалій трафіку збирає базову інформацію для розуміння нормальної роботи мережі, куди входять:

– Інтенсивність пакетів для кожного типу пакетів, обмірювана як кількість пакетів у секунду (pps).

– Співвідношення пакетів, наприклад, співвідношення пакетів SYN і пакетів FIN.

– Кількість одночасних TCP-з'єднань, відкритих одним джерелом.

КБПЗ_2025

Базова інформація збирається по кожній цільовій адресі хост-ПК, цільовій підмережі, вихідній адресі хост-ПК і вихідній підмережі.

Після закінчення періоду навчання Детектор аномалій трафіку переводиться в режим моніторингу, а Блок усунення аномального трафіку – у резервний режим готовності. Доти, поки немає атаки, що активно розвивається, вхідний трафік з мережі Інтернет проходить через комутатор без якого-небудь втручання з боку Блоку усунення аномального трафіку. Копія вхідного трафіку посилає для аналізу на Детектор аномалій трафіку через зовнішній аналізатор протоколів (SPAN) або віртуальні списки ACL. Якщо Детектор аномалій трафіку виявляє аномальне в порівнянні з базовою інформацією поводження трафіку, починається процес усунення:

– Детектор аномалій трафіку направляє в Блок усунення аномального трафіку команду почати процес зміни напрямку.

– Блок усунення аномального трафіку відхиляє (“захоплює”) трафік, адресований на атакуєму IP-адресу, переадресуючи його на самого себе.

– Блок усунення аномального трафіку піддає трафік багатоступінчастому аналізу й застосовує контрзаходи для відділення благонадійних джерел від джерел атаки. Цей процес іменується очищенням або вичищенням.

– Блок усунення аномального трафіку скидає трафік атаки й пересилає благонадійний трафік назад на нормальний маршрут проходження трафіку до мети. Цей процес іменується ін'єкцією.

Детектор аномалій трафіку

Детектор аномалій трафіку – це пасивний пристрій моніторингу, що постійно виявляє ознаки, що вказують на присутність атаки DDoS, спрямованої проти захищеного місця призначення, також іменованого зоною. Це може бути сервер, інтерфейс міжмережного екрана або інтерфейс маршрутизатора. Детектор аномалій трафіку аналізує копії всього вхідного трафіку, адресуємого в захищені зони, через SPAN або відгалуження пасивної мережі. Цей аналіз включає зіставлення поточного поводження трафіку з базовими граничними параметрами,

КОПЗ-2025

які також іменуються зональною політикою, для виявлення аномального поводження трафіку. Якщо аномальне поводження виявлене й виглядає як можлива атака, Детектор аномалій трафіку через позаполосну управлінську мережу Ethernet посилає в Блок усунення аномального трафіку сигнал про початок аналізу й усунення атаки.

Блок усунення аномального трафіку

Блок усунення аномального трафіку – це автономний пристрій аналізу й фільтрації трафіку. Починаючи прийом трафіку, адресованого в конкретну зону, що, очевидно, піддається атаці, Блок усунення аномального трафіку проводить точний аналіз цього трафіку. Якщо результати аналізу підтверджують, що трафік злочинний, Блок усунення аномального трафіку застосовує контрзаходи, наприклад, механізми анти-спуфінга й фільтрацію різного рівня (таблиця 3.1). Кінцевий результат полягає в тому, що трафік зі злочинних джерел скидається, а трафік із благонадійних джерел пересилається в передбачений пункт призначення.

Можливі варіанти зміни напрямку трафіку

Фахівці з IT можуть використовувати описані нижче варіанти зміни напрямку трафіку з його пересиланням з мережі, розташованого вище лежачого оператора зв'язку, на Блок усунення аномального трафіку. Цей процес також іменується “захватом” трафіку:

– Повідомлення прикордонного шлюзового протоколу (Border Gateway Protocol, BGP) із Блок усунення аномального трафіку на маршрутизатори, розташовані у вище лежачого оператора зв'язку, з інформацією про те, що трафік, адресований на захищену адресу призначення, буде переспрямований на Блок усунення аномального трафіку.

– Використання зовнішніх механізмів зміни напрямку трафіку, наприклад, маршрутизаторів віддаленого відновлення BGP.

– Повідомлення про ін'єкцію очищеного трафіку на маршруті (Route Health Injection, RHI) від Блок усунення аномального трафіку для процесу

КПЗ-2025

маршрутизації в Catalyst серії 6500 або в систему нагляду серії 7600. Ці повідомлення поміщають статичний маршрут у глобальну таблицю маршрутизації, у якій модуль Блок усунення аномального трафіку позначений як наступний вузол.

Атаки DDoS – Виявлення й усунення

У таблиці 3.1 перераховані типи атак DDoS, які може виявляти й усувати Блок усунення аномального трафіку.

Таблиця 3.1 – Категорії й особливі типи атак DDoS

Категорія атаки	Особливі типи атак
Атаки із заповненням смуги пропущення	Лавинні атаки зі спуфінгом або без спуфінга: – Прапор TCP (SYN, SYN-ACK, ACK, FIN). – Протокол керування повідомленнями в Інтернет (ICMP). – Протокол користувальницьких датаграмм (UDP). Приклади: лавинна атака SYN, smurf, LAND і UDP – лавинні атаки.
	Атаки зомбі-комп'ютерів/мереж зомбі-комп'ютерів, у яких кожний вихідний зомбі-ПК або мережа відкриває множинні TCP-з'єднання й, у деяких випадках, видає багаторазові запити HTTP.
	Атаки DNS, наприклад, лавинна атака із запитами DNS.
Атаки з дефіцитом ресурсів	Атаки пакетного розміру, характерна риса яких – фрагментированні або великі пакети. Приклади: teardrop і ping-of-death.
	Атаки зомбі-комп'ютерів/мереж зомбі-комп'ютерів з низькою інтенсивністю схожі на атаки із заповненням смуги пропущення за тим виключенням, що кожне джерело атаки посилає множинні запити з невеликим обсягом в одиницю часу.
	Атаки DNS з рекурсивним переглядом DNS.

КПЗ-2025

Можливі варіанти ін'єкції трафіку

Ін'єкція трафіку – це процес, застосовуваний у Блок усунення аномального трафіку для пересилання очищеного благонадійного трафіку в точку призначення, що піддається атаці. Рішення підтримує різні варіанти ін'єкції трафіку. У варіанті 2-ого рівня топології, очищений трафік пересилається із Блок усунення аномального трафіку на статично-конфігуруєму наступну адресу заходу. Ця адреса перебуває на маршрутизаторі, розташованому нижче й з'єднаним з тої ж VLAN або підмережею, що й інтерфейс/VLAN ін'єкції трафіку. Ін'єкцію трафіку на 2-му рівні найпростіше конфігурувати, оскільки тут не потрібно вносити які-небудь істотні зміни в конфігурацію маршрутизатора, розташованого нижче.

Варіанти ін'єкції трафіку 3-го рівня:

– Маршрутизація й пересилання по VPN (VPN Routing and Forwarding, VRF).

– Маршрутизація на основі політики (Policy-Based Routing, PBR).

– Транкинг VLAN (VLAN Trunking).

– Інкапсуляція по загальній маршрутизації (GRE) або інкапсуляція IP у тунелі IP (IPIP).

Блок визначення виду атаки

Блок визначення виду атаки:

– Атака ARP-spoofing на таблицю mac-адрес комутаторів.

– Широкомовний шторм.

– Додатки, що роблять інтенсивне широкомовне розсилання, наприклад: широкомовні чати й мережні ігри.

КОПЗ-2025

Рисунок 3.2 – Функціональна схема системи

КПЗ 2025

ARP-spoofing

ARP-spoofing – техніка атаки в Ethernet мережах, що дозволяє перехоплювати трафік між хостами. Заснована на використанні протоколу ARP.

При використанні в розподіленій обчислювальній системи (РВМ) алгоритмів віддаленого пошуку існує можливість здійснення в такій мережі типової віддаленої атаки «помилковий об'єкт РВМ». Аналіз безпеки протоколу ARP показує, що, перехопивши на атакуючому хості усередині даного сегмента мережі широкомовний ARP-запит, можна послати помилкову ARP-відповідь, у якій оголосити себе шуканим хостом (наприклад, маршрутизатором), і надалі активно контролювати мережний трафік дезінформованного хосту, впливаючи на нього за схемою «помилковий об'єкт РВМ».

Протокол ARP призначений для перетворення IP-адрес в MAC-адреси. Найчастіше мова йде перетворенні в адреси Ethernet, але ARP використовується й у мережах інших технологій: Token Ring, FDDI і інших.

Алгоритм роботи ARP

Протокол може використовуватися в наступних випадках:

1. Хост А хоче передати IP-пакет вузлу В, що перебуває з ним в одній мережі.

2. Хост А хоче передати IP-пакет вузлу В, що перебуває з ним у різних мережах, і користується для цього послугами маршрутизатора R.

У кожному із цих випадку вузлом А буде використовуватися протокол ARP, тільки в першому випадку для визначення MAC-адреси вузла В, а в другому – для визначення MAC-адреси маршрутизатора R. В останньому випадку пакет буде переданий маршрутизатору для подальшої ретрансляції.

Далі для простоти розглядається перший випадок, коли інформацією обмінюються вузли, що перебувають безпосередньо в одній мережі. (Випадок коли пакет адресований вузлу, який знаходиться за маршрутизатором, відрізняється тільки тим, що в пакетах переданих після того як ARP-перетворення завершене, використовується IP-адреса одержувача, але MAC-адреса маршрутизатора, а не одержувача.)

КБПЗ-2025

Проблеми ARP

Протокол ARP є абсолютно незахищеним. Він не має ніякого способу перевірки дійсності пакетів: як запитів, так і відповідей. Ситуація стає ще більш складною, коли може використовуватися мимовільний ARP (gratuitous ARP).

Мимовільний ARP – таке поводження ARP, коли ARP-відповідь надсилається, коли в цьому (з погляду одержувача) немає особою необхідності. Мимовільна ARP-відповідь це пакет-відповідь ARP, присланий без запиту. Він застосовується для визначення конфліктів IP-адрес у мережі: як тільки станція одержує адресу по DHCP або адреса привласнюється вручну, розсилається ARP-відповідь gratuitous ARP.

Мимовільний ARP може бути корисний у наступних випадках:

– Відновлення ARP-таблиць, зокрема, у кластерних системах.

– Інформування комутаторів.

– Повідомлення про включення мережного інтерфейсу.

Незважаючи на ефективність мимовільного ARP, він є особливо небезпечним, оскільки з його допомогою можна запевнити віддалений вузол у тому, що MAC-адреса якої-небудь системи, що перебуває з нею в одній мережі, змінилася й указати, яка адреса використовується тепер.

До виконання ARP-spoofing'a в ARP-таблиці вузлів A і B існують записи з IP- і MAC-адресами один одного. Обмін інформацією виробляється безпосередньо між вузлами A і B.

У ході виконання ARP-spoofing'a комп'ютер **C**, що виконує атаку, відправляє ARP-відповіді (без одержання запитів):

– вузлу A: з IP-адресою вузла B і MAC-адресою вузла C;

– вузлу B: з IP-адресою вузла A і MAC-адресою вузла C.

У силу того що комп'ютери підтримують мимовільний ARP (gratuitous ARP), вони модифікують власні ARP-таблиці й поміщають туди записи, де замість справжніх MAC-адрес комп'ютерів A і B коштує MAC-адреса комп'ютера C.

КПІЗ_2025

Після того як атака виконана, коли комп'ютер А хоче передати пакет комп'ютеру В, він знаходить в ARP-таблиці запис (він відповідає комп'ютеру С) і визначає з її MAC-адресу одержувача. Відправлений по цьому MAC-адресу пакет приходить комп'ютеру С замість одержувача. Комп'ютер С потім ретранслює пакет тому, кому він дійсно адресований – тобто комп'ютеру В.

Широкомовний шторм

Широкомовний шторм – лавина (сплеск) широкомовних акетів (на другому рівні моделі OSI – кадрів). Розмноження некоректно сформованих широкомовних повідомлень у кожному вузлі приводить до експонентного росту їхнього числа й паралізує роботу мережі. Звичайно такі пакети використовуються мережними сервісами для оповіщення станцій про свою присутність. Вважається нормальним, якщо широкомовні пакети становлять не більше 10% від загального числа пакетів у мережі.

Також досить часто до шторму приводять кільця в мережі при некоректному настроюванні протоколу Spanning Tree, оскільки в заголовку пакетів Ethernet немає інформації про час життя кадру, як, наприклад, у пакетів IP. Крім цього широкомовний шторм застосовується (навмисно) зломщиками.

Відповідно до галузевого стандарту де-факто число широкомовних і багатоадресних кадрів у мережі не повинне перевищувати 8-10% від загального числа кадрів.

Широкомовний кадр – це кадр, адресований всім станціям у домені мережі. Багатоадресний кадр – це кадр, адресований групі станцій у домені мережі. Оскільки широкомовний кадр адресований всім станціям, то, одержавши його, станції повинні перервати свою роботу й обробити такий кадр. Це сповільнює роботу всієї мережі.

Якщо відношення числа широкомовних кадрів до загального числа кадрів більше 10%, то такий ефект називається "широкомовним штормом".

Широкомовний шторм може бути наслідком дефектів устаткування або неправильного настроювання параметрів активного встаткування.

Найчастіше це явище спостерігається в розподілених мережах NetWare, побудованих на основі комутаторів, або коли дані між сегментами або доменами

КОПЗ_2025

мережі можуть передаватися більш ніж по одному потенційному шляху. Якщо один з комутаторів такої мережі не підтримує протокол Spanning Tree (звичайно IEEE 802.1d) або останній неправильно настроєний або збоїть, то в мережі починається некерована циркуляція широкомовних кадрів.

Виявлення "широкомовного шторму" є не настільки тривіальним завданням, як це може здатися на перший погляд. Для його виявлення недостатньо взяти загальне число широкомовних кадрів і поділити його на загальне число кадрів, що пройшли по мережі.

Для цього ви повинні визначити: яку частку становлять широкомовні кадри в кожний інтервал часу (наприклад, за одну мінуту) і яка при цьому утилізація каналу зв'язку. Якщо, наприклад, за одну мінуту по мережі пройшло 4 кадри, а 2 з них були широкомовними, то це ще не виходить, що ви спостерігаєте "широкомовний шторм".

Захист від широкомовних штормів (broadcast storm)

Одна з характерних несправностей мережного програмного забезпечення – мимовільна генерація з високою інтенсивністю широкомовних пакетів. Широкомовним штормом вважається ситуація, у якій відсоток широкомовних пакетів перевищує 20% від загальної кількості пакетів у мережі. Звичайний комутатор або міст сліпо передає такі пакети на всі свої порти, як того вимагає його логіка роботи, засмічуючи, таким чином, мережу.

Боротьба із широкомовним штормом у мережі, з'єднаної комутаторами, жадає від адміністратора відключення портів, що генерують широкомовні пакети. Маршрутизатор не поширює такі ушкоджені пакети, оскільки в коло його завдань не входить копіювання широкомовних пакетів в усі поєднувані їм мережі.

Тому маршрутизатор є прекрасним засобом боротьби із широкомовним штормом, щоправда, якщо мережа розділена на достатню кількість підмереж.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

КОПІЯ 2025

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврської дипломної роботи, наведена на рисунку 3.3.

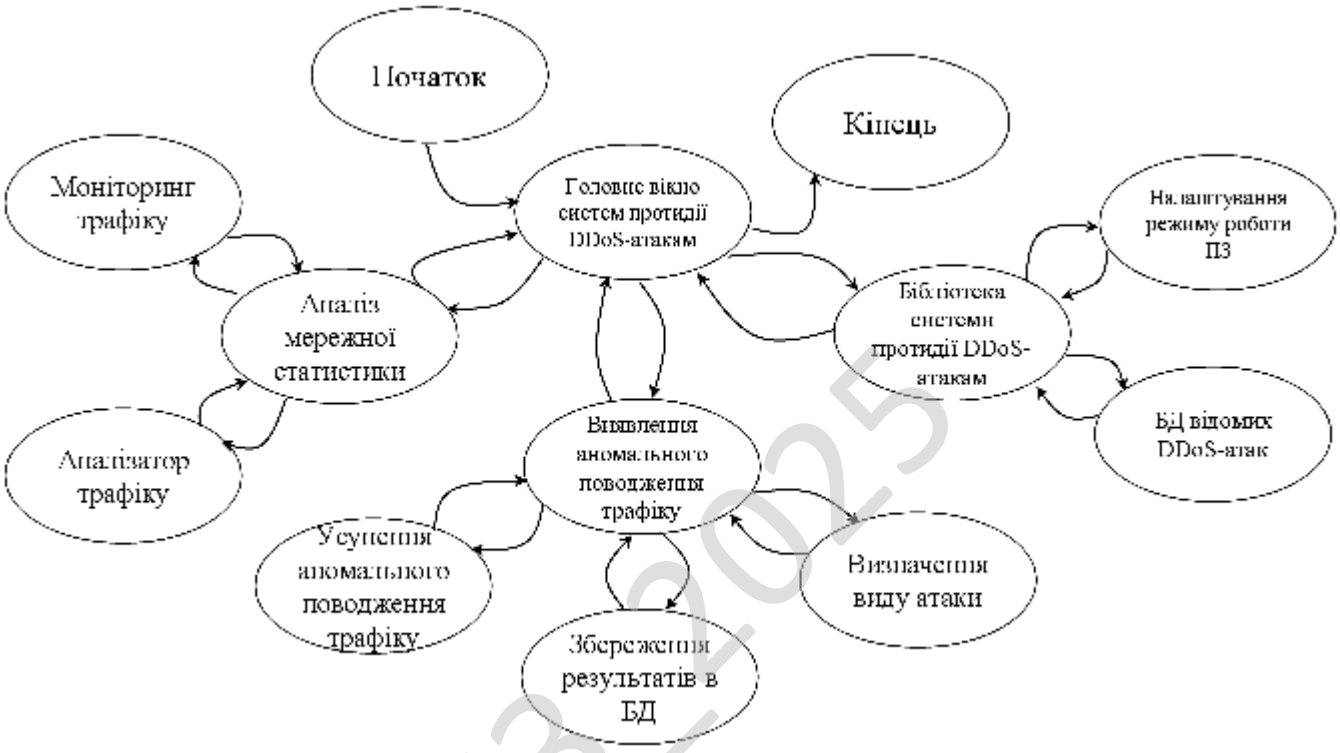

Рисунок 3.3 – Діаграма взаємодії процесів

КОПЗ 2025

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

КбПЗ_2025

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю систем протидії DDoS-атакам.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення,

КПЗ-2025

візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

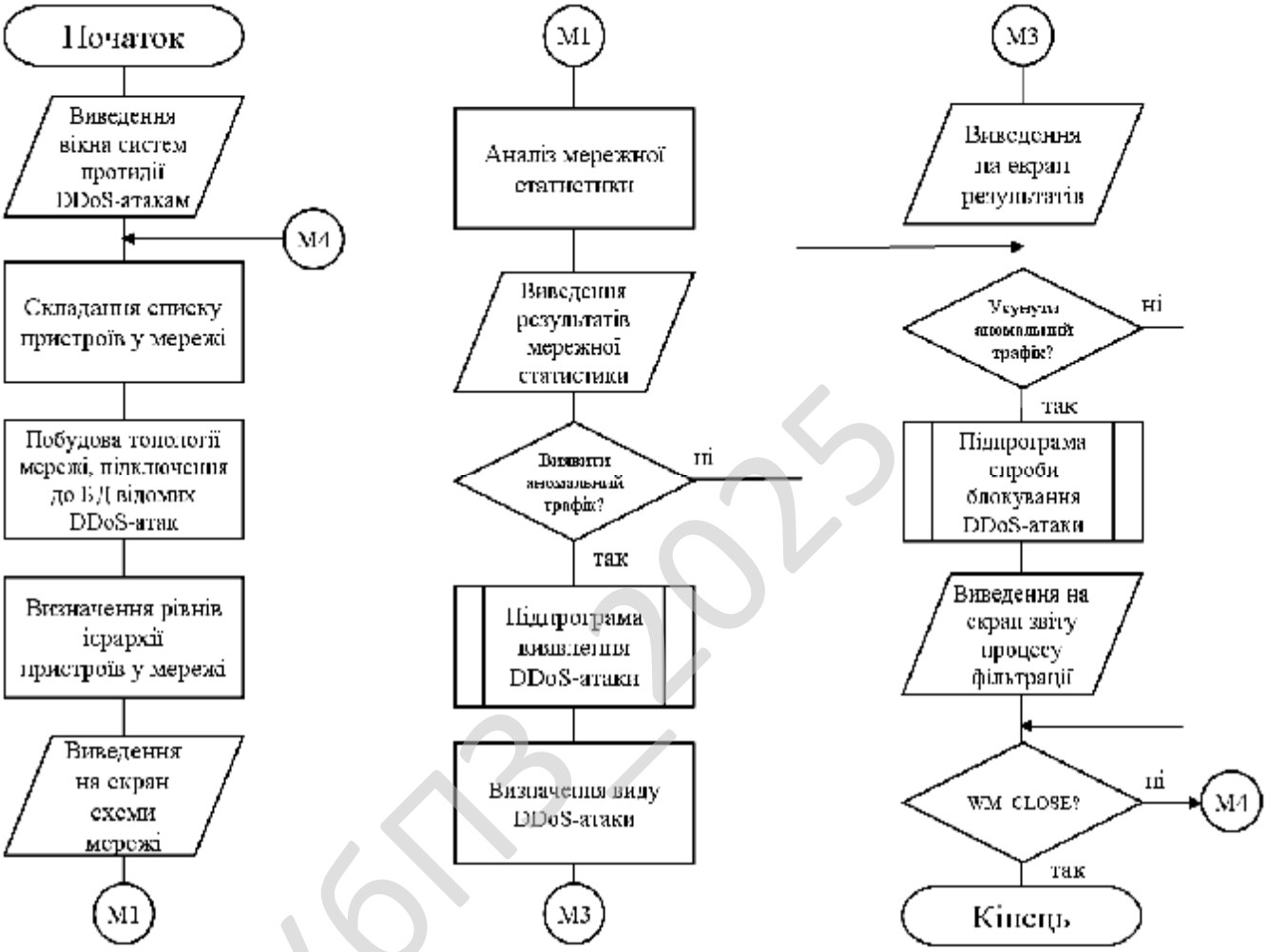

Рисунок 4.1 – Блок-схема основної програми

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

КбПЗ-2025

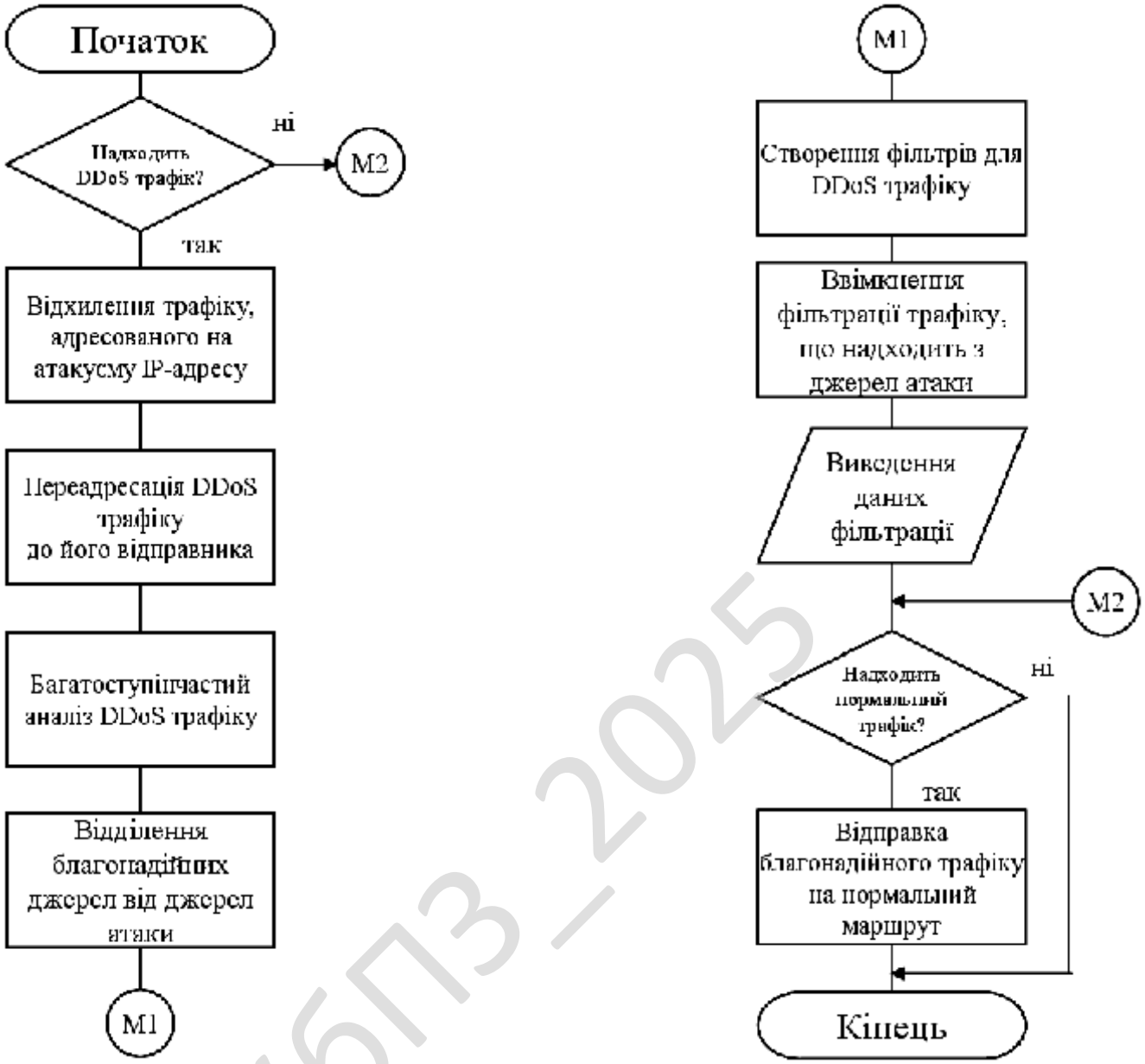

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

КбПЗ_2025

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.Дія є фундаментальною

КПЗ-2025

одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

КПЗ-2025

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого

КПЗ_2025

програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку

КПЗ-2025

можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням. Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Опис системи

Система протидії DDoS-атакам забезпечує виявлення, аналіз та блокування шкідливого трафіку. Для реалізації обираємо архітектуру на основі моніторингу мережевого трафіку з використанням методу аналізу поведінкових аномалій та порівняння із встановленими пороговими значеннями.

Основні компоненти системи:

1. Модуль моніторингу мережевого трафіку.
2. Модуль аналізу трафіку.
3. Модуль ухвалення рішень.
4. Модуль блокування шкідливого трафіку.
5. Модуль логування подій.

КПЗ-2025

Основні модулі системи (виділена частина коду)

1. Модуль моніторингу мережевого трафіку.

Відповідає за отримання вхідних даних з мережевого інтерфейсу та формування потоків даних для подальшого аналізу. Реалізується за допомогою бібліотеки scapy.

```
from scapy.all import sniff, IP
import time

class TrafficMonitor:
    def __init__(self):
        self.packet_count = 0
        self.start_time = time.time()

    def monitor_traffic(self, iface="eth0"):
        sniff(iface=iface, prn=self.process_packet, store=0)

    def process_packet(self, packet):
        if IP in packet:
            self.packet_count += 1
            print(f"Packet received from {packet[IP].src}")
```

2. Модуль аналізу трафіку

Аналізує вхідний трафік для виявлення аномалій. Використовує алгоритм визначення порогових значень за швидкістю отримання пакетів та обчислює середню швидкість.

```
class TrafficAnalyzer:
    def __init__(self, threshold=100):
        self.threshold = threshold
# Поріг кількості пакетів за секунду
    def analyze(self, monitor):
        elapsed_time = time.time() - monitor.start_time
        packet_rate = monitor.packet_count / elapsed_time
        print(f"Packet rate: {packet_rate:.2f} packets/sec")
        if packet_rate > self.threshold:
            print("Potential DDoS attack detected")
            return True
        return False
```

КОПЗ_2025

3. Модуль ухвалення рішень

Приймає рішення щодо блокування трафіку на основі даних з модуля аналізу.

```
class DecisionMaker:
    def __init__(self):
      pass
    def make_decision(self, is_attack):
        if is_attack:
            print("Blocking traffic due to DDoS attack")
            return True
        print("Traffic is normal")
        return False
```

4. Модуль блокування шкідливого трафіку

Забезпечує блокування виявленого шкідливого трафіку. Реалізується за допомогою взаємодії з iptables.

```
import os

class TrafficBlocker:
    def __init__(self):
        pass

    def block_traffic(self):
        os.system("iptables -A INPUT -j DROP")
        print("All incoming traffic blocked")
```

5. Модуль логування подій

Фіксує події в системі для подальшого аналізу та аудиту.

```
class Logger:
    def __init__(self, log_file="system_log.txt"):
        self.log_file = log_file

    def log_event(self, message):
        with open(self.log_file, "a") as file:
            file.write(f"{time.ctime()} - {message}\n")
```

КОПЗ_2025

Архітектура системи

Система працює за принципом послідовного аналізу. Модуль моніторингу передає дані модулю аналізу. Після цього модуль ухвалення рішень визначає необхідність блокування трафіку і передає відповідну команду до модуля блокування. Усі події реєструються модулем логування.

Розрахунки та обґрунтування вибору порогових значень

Поріг кількості пакетів встановлюється залежно від нормального трафіку мережі. Наприклад, середня кількість пакетів у звичайному режимі роботи складає 50 пакетів за секунду. Для запобігання помилковому спрацьовуванню встановлюємо поріг у 100 пакетів за секунду. Це дає змогу вчасно виявити аномалії та мінімізувати ризик DDoS-атаки.

Основний код для запуску системи.

```
if __name__ == "__main__":
    monitor = TrafficMonitor()
    analyzer = TrafficAnalyzer()
    decision_maker = DecisionMaker()
    blocker = TrafficBlocker()
    logger = Logger()
    monitor.monitor_traffic()
    is_attack = analyzer.analyze(monitor)
    decision = decision_maker.make_decision(is_attack)

    if decision:
        blocker.block_traffic()
        logger.log_event("DDoS attack detected and traffic blocked")
    else:
        logger.log_event("Traffic analyzed. No attack detected")
```

Ця система надає базову функціональність для моніторингу мережі та виявлення DDoS-атак. Для підвищення ефективності рекомендується додати механізм автоматичного зняття блокування після певного часу та використовувати додаткові методи машинного навчання для покращення точності виявлення атак.

КбПЗ_2025

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою алгоритму захисту інформації RSA. Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток $N = P*Q$ і значення функції $\varphi(N) = (P-1)(Q-1)$. Далі відправник обчислює число E з умов $E < \varphi(N)$, НЗД $(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E*D \equiv 1 (\text{mod } \varphi(N))$.

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: $m = h(M)$.

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m \, (\text{mod } N)$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E \, (\text{mod } N)$.

Крім того, він знаходить результат гешування прийнятого повідомлення M с допомогою такої ж геш-функції h(-): $m = h(M)$.

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (M, S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти

КБПЗ-2025

модуль N на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому випадку, якщо при обчисленні S був використаний секретний ключ D, що відповідає відкритому ключу E. Тому відкритий ключ E іноді називають "ідентифікатором" того, хто підписав.

КбПЗ_2025

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення систем протидії DDoS-атакам складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Вид; Інструменти; Параметри; Довідка.

– Підрозділу представлення інформаційних даних.

– Вікна обрання типу.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Графічних функціональних кнопок ПЗ.

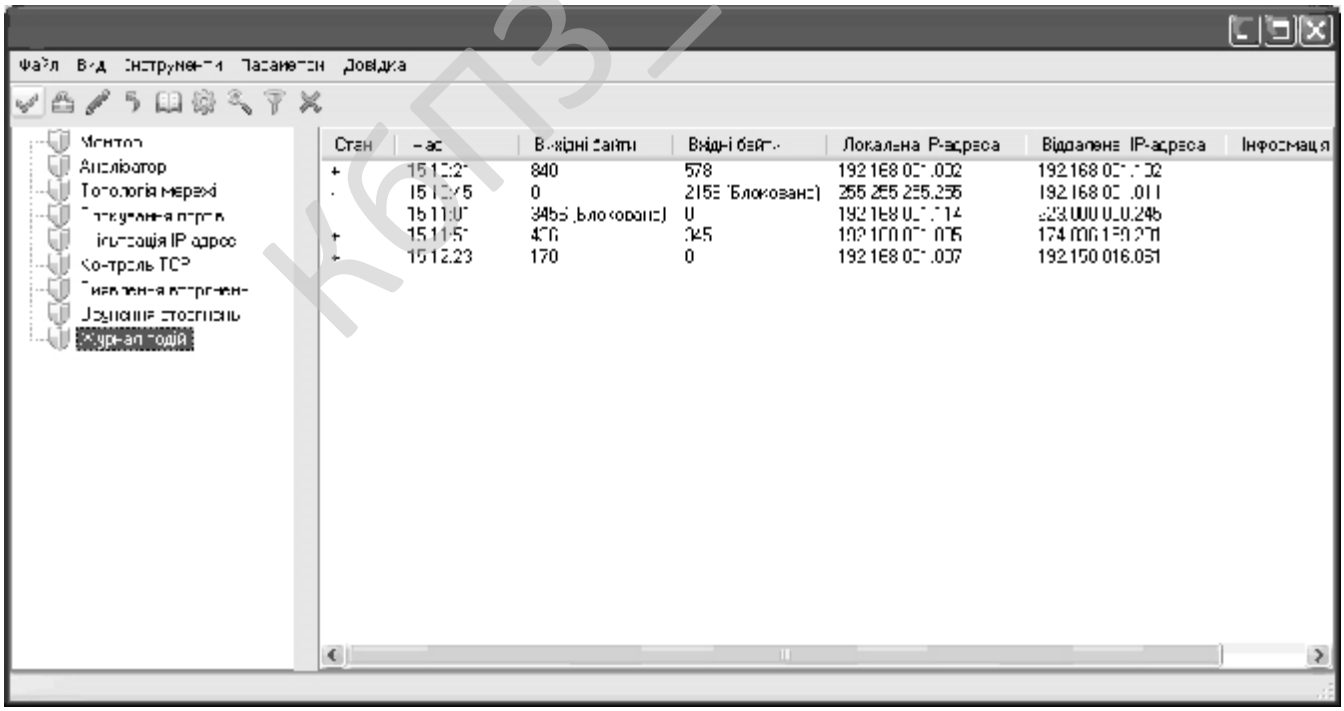

Рисунок 5.1 – Головне вікно розробленого ПЗ

КПЗ – 2025

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

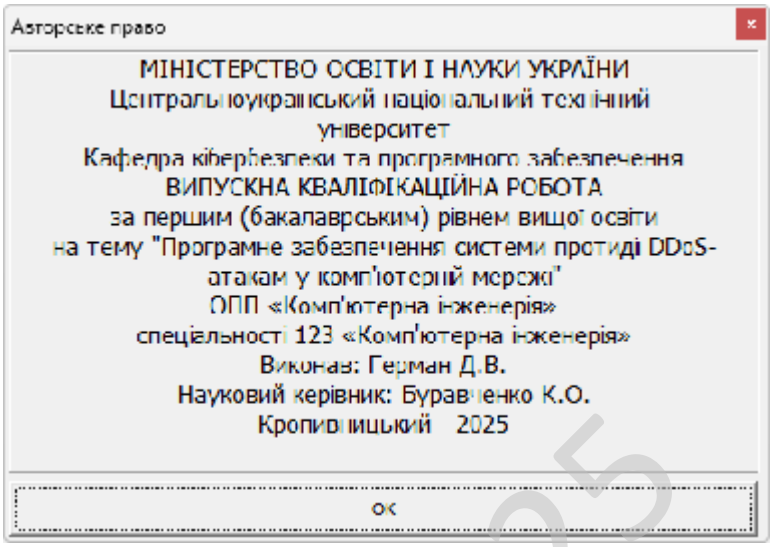

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

КОПІЯ 2025

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

КПЗ-2025

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

КПЗ-2025

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи протидії DDoS-атакам у комп'ютерній мережі.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем протидії DDoS-атакам у комп'ютерній мережі.

– Досліджена система протидії DDoS-атакам у комп'ютерній мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи протидії DDoS-атакам у комп'ютерній мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання протидії DDoS-атакам у комп'ютерній мережі.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи протидії DDoS-атакам у комп'ютерній мережі. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної

КПЗ-2025

техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RSA.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

КбПЗ_2025

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing Ltd. 2022. 573 p.

2. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

3. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

4. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

5. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

6. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

7. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

8. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

9. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

10. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

11. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

12. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

13. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

14. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted

КБПЗ_2025

Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

15. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

16. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

17. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

18. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

19. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

20. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

21. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

КОПЗ_2025

22. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

23. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

24. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

25. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

26. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

27. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

28. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

КОПЗ-2025

29. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

30. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

31. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

32. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

33. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

34. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

35. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

36. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

КБПЗ-2025

37. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

38. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

39. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

40. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

41. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

42. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

43. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

44. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

КОПЗ_2025

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

45. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

46. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

47. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

48. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

49. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

50. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

КПІ3_2025

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Перелік документів, що розробляються.....5
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

КПЗ–2025

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи протидії DDoS-атакам у комп’ютерній мережі.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 48-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи протидії DDoS-атакам у комп’ютерній мережі.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

КБПЗ_2025

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи протидії DDoS-атакам у комп’ютерній мережі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КПЗ_2025

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

КбПЗ – 2025

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 76 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

КБПЗ_2025

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

КбПЗ_2025

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

Програмне забезпечення системи протидії DDoS-атакам у комп’ютерній мережі

КбПЗ_2025

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 23

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import os
import time
from collections import defaultdict
from scapy.all import sniff, IP

class TrafficMonitor:
    def __init__(self):
        self.packet_count = defaultdict(int)
        self.start_time = time.time()

    def monitor_traffic(self, iface="eth0"):
        sniff(iface=iface, prn=self.process_packet, store=0)

    def process_packet(self, packet):
        if IP in packet:
            src_ip = packet[IP].src
            self.packet_count[src_ip] += 1
            print(f"Packet received from {src_ip}")

class TrafficAnalyzer:
    def __init__(self, threshold_per_ip=50, global_threshold=1000):
        self.threshold_per_ip = threshold_per_ip
        self.global_threshold = global_threshold

    def analyze(self, monitor):
        current_time = time.time()
        elapsed_time = current_time - monitor.start_time

        per_ip_violations = {}
        total_packets = sum(monitor.packet_count.values())

        for ip, count in monitor.packet_count.items():
            packet_rate = count / elapsed_time

            if packet_rate > self.threshold_per_ip:
                per_ip_violations[ip] = packet_rate

        if total_packets / elapsed_time > self.global_threshold:
            print("Global packet rate exceeded threshold")
            return per_ip_violations, True

        print("Traffic within normal limits")
        return per_ip_violations, False

class DecisionMaker:
    def __init__(self):
        pass

    def make_decision(self, per_ip_violations, global_attack):
        if global_attack:
            print("Global DDoS attack detected. Blocking all traffic")
            return True, []

        if per_ip_violations:
```

КБПЗ_2025

```
            print(f"Per-IP DDoS attack detected from:
{list(per_ip_violations.keys())}")
            return False, list(per_ip_violations.keys())

        print("Traffic is normal")
        return False, []

class TrafficBlocker:
    def __init__(self):
        pass

    def block_ip(self, ip):
        os.system(f"iptables -A INPUT -s {ip} -j DROP")
        print(f"Blocked IP: {ip}")

    def block_all(self):
        os.system("iptables -A INPUT -j DROP")
        print("Blocked all incoming traffic")

class Logger:
    def __init__(self, log_file="system_log.txt"):
        self.log_file = log_file

    def log_event(self, message):
        with open(self.log_file, "a") as file:
            file.write(f"{time.ctime()} - {message}\n")

class MitigationManager:
    def __init__(self, cooldown_time=300):
        self.cooldown_time = cooldown_time
        self.blocked_ips = {}

    def manage_cooldown(self):
        current_time = time.time()
        to_unblock = [ip for ip, timestamp in self.blocked_ips.items() if
current_time - timestamp > self.cooldown_time]

        for ip in to_unblock:
            os.system(f"iptables -D INPUT -s {ip} -j DROP")
            print(f"Unblocked IP: {ip}")
            del self.blocked_ips[ip]

    def block_ip_with_cooldown(self, ip):
        self.blocked_ips[ip] = time.time()
        os.system(f"iptables -A INPUT -s {ip} -j DROP")
        print(f"Blocked IP with cooldown: {ip}")

if __name__ == "__main__":
    monitor = TrafficMonitor()
    analyzer = TrafficAnalyzer()
    decision_maker = DecisionMaker()
    blocker = TrafficBlocker()
    logger = Logger()
    mitigation_manager = MitigationManager()

    try:
        print("Starting DDoS protection system...")
```

КОПЗ_2025

```
        monitor.monitor_traffic()

        while True:
            mitigation_manager.manage_cooldown()
            per_ip_violations, global_attack = analyzer.analyze(monitor)
            global_decision, ips_to_block =
decision_maker.make_decision(per_ip_violations, global_attack)

            if global_decision:
                blocker.block_all()
                logger.log_event("Global DDoS attack detected. All traffic
blocked")
            else:
                for ip in ips_to_block:
                    mitigation_manager.block_ip_with_cooldown(ip)
                    logger.log_event(f"DDoS attack detected from {ip}. Traffic
blocked")

            time.sleep(5)

    except KeyboardInterrupt:
        print("Stopping DDoS protection system...")
```

КбПЗ_2025

Файл start_traffic_collection.py

```
import switch
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub

from datetime import datetime

# class CollectTrainingStatsApp(simple_switch_13.SimpleSwitch13):
class CollectTrainingStatsApp(switch.SimpleSwitch13):
    def __init__(self, *args, **kwargs):
        super(CollectTrainingStatsApp, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self.monitor)

        file0 = open("FlowStatsfile.csv","w")

file0.write('timestamp,datapath_id,flow_id,ip_src,tp_src,ip_dst,tp_dst,ip_proto,
icmp_code,icmp_type,flow_duration_sec,flow_duration_nsec,idle_timeout,hard_timeo
ut,flags,packet_count,byte_count,packet_count_per_second,packet_count_per_nsecon
d,byte_count_per_second,byte_count_per_nsecond,label\n')
        file0.close()

    #Asynchronous message
    @set_ev_cls(ofp_event.EventOFPStateChange,[MAIN_DISPATCHER,
DEAD_DISPATCHER])
    def state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                self.datapaths[datapath.id] = datapath

        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                del self.datapaths[datapath.id]


    def monitor(self):
        while True:
            for dp in self.datapaths.values():
                self.request_stats(dp)
            hub.sleep(10)


    def request_stats(self, datapath):
        self.logger.debug('send stats request: %016x', datapath.id)

        parser = datapath.ofproto_parser

        req = parser.OFPFlowStatsRequest(datapath)
        datapath.send_msg(req)

    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
```

КБГЗ_2025

```
    def _flow_stats_reply_handler(self, ev):

        timestamp = datetime.now()
        timestamp = timestamp.timestamp()
        icmp_code = -1
        icmp_type = -1
        tp_src = 0
        tp_dst = 0

        file0 = open("FlowStatsfile.csv","a+")
        body = ev.msg.body
        for stat in sorted([flow for flow in body if (flow.priority == 1) ],
key=lambda flow:

(flow.match['eth_type'],flow.match['ipv4_src'],flow.match['ipv4_dst'],flow.match
['ip_proto'])):


            ip_src = stat.match['ipv4_src']
            ip_dst = stat.match['ipv4_dst']
            ip_proto = stat.match['ip_proto']

            if stat.match['ip_proto'] == 1:
                icmp_code = stat.match['icmpv4_code']
                icmp_type = stat.match['icmpv4_type']

            elif stat.match['ip_proto'] == 6:
                tp_src = stat.match['tcp_src']
                tp_dst = stat.match['tcp_dst']

            elif stat.match['ip_proto'] == 17:
                tp_src = stat.match['udp_src']
                tp_dst = stat.match['udp_dst']

            flow_id = str(ip_src) + str(tp_src) + str(ip_dst) + str(tp_dst) +
str(ip_proto)

            try:
                packet_count_per_second = stat.packet_count/stat.duration_sec
                packet_count_per_nsecond = stat.packet_count/stat.duration_nsec
            except:
                packet_count_per_second = 0
                packet_count_per_nsecond = 0

            try:
                byte_count_per_second = stat.byte_count/stat.duration_sec
                byte_count_per_nsecond = stat.byte_count/stat.duration_nsec
            except:
                byte_count_per_second = 0
                byte_count_per_nsecond = 0



file0.write("{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}\n
"
                .format(timestamp, ev.msg.datapath.id, flow_id, ip_src,
tp_src,ip_dst, tp_dst,
```

КОПИЯ 2025

```
                    stat.match['ip_proto'],icmp_code,icmp_type,
                    stat.duration_sec, stat.duration_nsec,
                    stat.idle_timeout, stat.hard_timeout,
                    stat.flags, stat.packet_count,stat.byte_count,
                    packet_count_per_second,packet_count_per_nsecond,
                    byte_count_per_second,byte_count_per_nsecond,0))
file0.close()
```

КбПЗ_2025

Файл mitigation_module.py.py

```
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub

import switchm
from datetime import datetime

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

class SimpleMonitor13(switchm.SimpleSwitch13):

    def __init__(self, *args, **kwargs):

        super(SimpleMonitor13, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self._monitor)

        start = datetime.now()

        self.flow_training()

        end = datetime.now()
        print("Training time: ", (end-start))

    @set_ev_cls(ofp_event.EventOFPStateChange,
                [MAIN_DISPATCHER, DEAD_DISPATCHER])
    def _state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                self.datapaths[datapath.id] = datapath
        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                del self.datapaths[datapath.id]

    def _monitor(self):
        while True:
            for dp in self.datapaths.values():
                self._request_stats(dp)
            hub.sleep(10)

            self.flow_predict()

    def _request_stats(self, datapath):
        self.logger.debug('send stats request: %016x', datapath.id)
        parser = datapath.ofproto_parser
```

КоТЗ - 2025

```
        req = parser.OFPFlowStatsRequest(datapath)
        datapath.send_msg(req)

    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _flow_stats_reply_handler(self, ev):

        timestamp = datetime.now()
        timestamp = timestamp.timestamp()

        file0 = open("PredictFlowStatsfile.csv","w")

file0.write('timestamp,datapath_id,flow_id,ip_src,tp_src,ip_dst,tp_dst,ip_proto,
icmp_code,icmp_type,flow_duration_sec,flow_duration_nsec,idle_timeout,hard_timeo
ut,flags,packet_count,byte_count,packet_count_per_second,packet_count_per_nsecon
d,byte_count_per_second,byte_count_per_nsecond\n')
        body = ev.msg.body
        icmp_code = -1
        icmp_type = -1
        tp_src = 0
        tp_dst = 0

        for stat in sorted([flow for flow in body if (flow.priority == 1) ],
key=lambda flow:

(flow.match['eth_type'],flow.match['ipv4_src'],flow.match['ipv4_dst'],flow.match
['ip_proto'])):

            ip_src = stat.match['ipv4_src']
            ip_dst = stat.match['ipv4_dst']
            ip_proto = stat.match['ip_proto']

            if stat.match['ip_proto'] == 1:
                icmp_code = stat.match['icmpv4_code']
                icmp_type = stat.match['icmpv4_type']

            elif stat.match['ip_proto'] == 6:
                tp_src = stat.match['tcp_src']
                tp_dst = stat.match['tcp_dst']

            elif stat.match['ip_proto'] == 17:
                tp_src = stat.match['udp_src']
                tp_dst = stat.match['udp_dst']

            flow_id = str(ip_src) + str(tp_src) + str(ip_dst) + str(tp_dst) +
str(ip_proto)

            try:
                packet_count_per_second = stat.packet_count/stat.duration_sec
                packet_count_per_nsecond = stat.packet_count/stat.duration_nsec
            except:
                packet_count_per_second = 0
                packet_count_per_nsecond = 0

            try:
                byte_count_per_second = stat.byte_count/stat.duration_sec
                byte_count_per_nsecond = stat.byte_count/stat.duration_nsec
            except:
```

КОПИЯ 2025

```
                byte_count_per_second = 0
                byte_count_per_nsecond = 0


file0.write("{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}\n"
                .format(timestamp, ev.msg.datapath.id, flow_id, ip_src,
tp_src,ip_dst, tp_dst,
                        stat.match['ip_proto'],icmp_code,icmp_type,
                        stat.duration_sec, stat.duration_nsec,
                        stat.idle_timeout, stat.hard_timeout,
                        stat.flags, stat.packet_count,stat.byte_count,
                        packet_count_per_second,packet_count_per_nsecond,
                        byte_count_per_second,byte_count_per_nsecond))

        file0.close()

    def flow_training(self):

        self.logger.info("Flow Training ...")

        flow_dataset = pd.read_csv('FlowStatsfile.csv')

        flow_dataset.iloc[:, 2] = flow_dataset.iloc[:, 2].str.replace('.', '')
        flow_dataset.iloc[:, 3] = flow_dataset.iloc[:, 3].str.replace('.', '')
        flow_dataset.iloc[:, 5] = flow_dataset.iloc[:, 5].str.replace('.', '')

        X_flow = flow_dataset.iloc[:, :-1].values
        X_flow = X_flow.astype('float64')

        y_flow = flow_dataset.iloc[:, -1].values

        X_flow_train, X_flow_test, y_flow_train, y_flow_test =
train_test_split(X_flow, y_flow, test_size=0.25, random_state=0)

        classifier = RandomForestClassifier(n_estimators=10,
criterion="entropy", random_state=0)
        self.flow_model = classifier.fit(X_flow_train, y_flow_train)

        y_flow_pred = self.flow_model.predict(X_flow_test)
        #y_flow_pred_train = self.flow_model.predict(X_flow_train)

self.logger.info("------------------------------------------------------------------
----------------")

        self.logger.info("Confusion Matrix")
        cm = confusion_matrix(y_flow_test, y_flow_pred)
        self.logger.info(cm)

       #acc = accuracy_score(y_flow_test, y_flow_pred)

        acc_train = accuracy_score(y_flow_train, y_flow_pred_train)
        print("Training Accuracy: ",acc_train)

        self.logger.info("Succes Accuracy = {0:.2f} %".format(acc*100))
        fail = 1.0 - acc
        self.logger.info("Fail Accuracy = {0:.2f} %".format(fail*100))
```

KGTS_2025

```
self.logger.info("------------------------------------------------------------
---------------")

    def flow_predict(self):
        try:
            predict_flow_dataset = pd.read_csv('PredictFlowStatsfile.csv')

            predict_flow_dataset.iloc[:, 2] = predict_flow_dataset.iloc[:,
2].str.replace('.', '')
            predict_flow_dataset.iloc[:, 3] = predict_flow_dataset.iloc[:,
3].str.replace('.', '')
            predict_flow_dataset.iloc[:, 5] = predict_flow_dataset.iloc[:,
5].str.replace('.', '')

            X_predict_flow = predict_flow_dataset.iloc[:, :].values
            X_predict_flow = X_predict_flow.astype('float64')

            y_flow_pred = self.flow_model.predict(X_predict_flow)

            legitimate_trafic = 0
            ddos_trafic = 0

            for i in y_flow_pred:
                if i == 0:
                    legitimate_trafic = legitimate_trafic + 1
                else:
                    ddos_trafic = ddos_trafic + 1
                    victim = int(predict_flow_dataset.iloc[i, 5])%20




            self.logger.info("------------------------------------------------
--------------------------")
            if (legitimate_trafic/len(y_flow_pred)*100) > 80:
                self.logger.info("Traffic is Legitimate!")
            else:
                self.logger.info("NOTICE!! DoS Attack in Progress!!!")
                self.logger.info("Victim Host: h{}".format(victim))
                print("Mitigation process in progress!")
                self.mitigation = 1

            self.logger.info("------------------------------------------------
--------------------------")

            file0 = open("PredictFlowStatsfile.csv","w")


file0.write('timestamp,datapath_id,flow_id,ip_src,tp_src,ip_dst,tp_dst,ip_proto,
icmp_code,icmp_type,flow_duration_sec,flow_duration_nsec,idle_timeout,hard_timeo
ut,flags,packet_count,byte_count,packet_count_per_second,packet_count_per_nsecon
d,byte_count_per_second,byte_count_per_nsecond\n')
            file0.close()

        except:
            pass
```

КОПЗ_2025

Файл controller.py

```
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub

import switch
from datetime import datetime

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

class SimpleMonitor13(switch.SimpleSwitch13):

    def __init__(self, *args, **kwargs):

        super(SimpleMonitor13, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self._monitor)

        start = datetime.now()

        self.flow_training()

        end = datetime.now()
        print("Training time: ", (end-start))

    @set_ev_cls(ofp_event.EventOFPStateChange,
                [MAIN_DISPATCHER, DEAD_DISPATCHER])
    def _state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                self.datapaths[datapath.id] = datapath
        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                del self.datapaths[datapath.id]

    def _monitor(self):
        while True:
            for dp in self.datapaths.values():
                self._request_stats(dp)
            hub.sleep(10)

            self.flow_predict()

    def _request_stats(self, datapath):
        self.logger.debug('send stats request: %016x', datapath.id)
        parser = datapath.ofproto_parser

        req = parser.OFPFlowStatsRequest(datapath)
```

КБП3_2025

```
        datapath.send_msg(req)

    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _flow_stats_reply_handler(self, ev):

        timestamp = datetime.now()
        timestamp = timestamp.timestamp()

        file0 = open("PredictFlowStatsfile.csv","w")

file0.write('timestamp,datapath_id,flow_id,ip_src,tp_src,ip_dst,tp_dst,ip_proto,
icmp_code,icmp_type,flow_duration_sec,flow_duration_nsec,idle_timeout,hard_timeo
ut,flags,packet_count,byte_count,packet_count_per_second,packet_count_per_nsecon
d,byte_count_per_second,byte_count_per_nsecond\n')
        body = ev.msg.body
        icmp_code = -1
        icmp_type = -1
        tp_src = 0
        tp_dst = 0

        for stat in sorted([flow for flow in body if (flow.priority == 1) ],
key=lambda flow:

(flow.match['eth_type'],flow.match['ipv4_src'],flow.match['ipv4_dst'],flow.match
['ip_proto'])):

            ip_src = stat.match['ipv4_src']
            ip_dst = stat.match['ipv4_dst']
            ip_proto = stat.match['ip_proto']

            if stat.match['ip_proto'] == 1:
                icmp_code = stat.match['icmpv4_code']
                icmp_type = stat.match['icmpv4_type']

            elif stat.match['ip_proto'] == 6:
                tp_src = stat.match['tcp_src']
                tp_dst = stat.match['tcp_dst']

            elif stat.match['ip_proto'] == 17:
                tp_src = stat.match['udp_src']
                tp_dst = stat.match['udp_dst']

            flow_id = str(ip_src) + str(tp_src) + str(ip_dst) + str(tp_dst) +
str(ip_proto)

            try:
                packet_count_per_second = stat.packet_count/stat.duration_sec
                packet_count_per_nsecond = stat.packet_count/stat.duration_nsec
            except:
                packet_count_per_second = 0
                packet_count_per_nsecond = 0

            try:
                byte_count_per_second = stat.byte_count/stat.duration_sec
                byte_count_per_nsecond = stat.byte_count/stat.duration_nsec
            except:
                byte_count_per_second = 0
```

KCTB_2025

```
                byte_count_per_nsecond = 0


file0.write("{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}\n"
                .format(timestamp, ev.msg.datapath.id, flow_id, ip_src,
tp_src,ip_dst, tp_dst,
                        stat.match['ip_proto'],icmp_code,icmp_type,
                        stat.duration_sec, stat.duration_nsec,
                        stat.idle_timeout, stat.hard_timeout,
                        stat.flags, stat.packet_count,stat.byte_count,
                        packet_count_per_second,packet_count_per_nsecond,
                        byte_count_per_second,byte_count_per_nsecond))

        file0.close()

    def flow_training(self):

        self.logger.info("Flow Training ...")

        flow_dataset = pd.read_csv('FlowStatsfile.csv')

        flow_dataset.iloc[:, 2] = flow_dataset.iloc[:, 2].str.replace('.', '')
        flow_dataset.iloc[:, 3] = flow_dataset.iloc[:, 3].str.replace('.', '')
        flow_dataset.iloc[:, 5] = flow_dataset.iloc[:, 5].str.replace('.', '')

        X_flow = flow_dataset.iloc[:, :-1].values
        X_flow = X_flow.astype('float64')

        y_flow = flow_dataset.iloc[:, -1].values

        X_flow_train, X_flow_test, y_flow_train, y_flow_test =
train_test_split(X_flow, y_flow, test_size=0.25, random_state=0)

        classifier = RandomForestClassifier(n_estimators=10,
criterion="entropy", random_state=0)
        self.flow_model = classifier.fit(X_flow_train, y_flow_train)

        y_flow_pred = self.flow_model.predict(X_flow_test)

        self.logger.info("------------------------------------------------------
------------------------")

        self.logger.info("confusion matrix")
        cm = confusion_matrix(y_flow_test, y_flow_pred)
        self.logger.info(cm)

        acc = accuracy_score(y_flow_test, y_flow_pred)

        self.logger.info("succes accuracy = {0:.2f} %".format(acc*100))
        fail = 1.0 - acc
        self.logger.info("fail accuracy = {0:.2f} %".format(fail*100))
        self.logger.info("------------------------------------------------------
------------------------")

    def flow_predict(self):
        try:
            predict_flow_dataset = pd.read_csv('PredictFlowStatsfile.csv')
```

КПЗ-2025

```
            predict_flow_dataset.iloc[:, 2] = predict_flow_dataset.iloc[:,
2].str.replace('.', '')
            predict_flow_dataset.iloc[:, 3] = predict_flow_dataset.iloc[:,
3].str.replace('.', '')
            predict_flow_dataset.iloc[:, 5] = predict_flow_dataset.iloc[:,
5].str.replace('.', '')

            X_predict_flow = predict_flow_dataset.iloc[:, :].values
            X_predict_flow = X_predict_flow.astype('float64')

            y_flow_pred = self.flow_model.predict(X_predict_flow)

            legitimate_trafic = 0
            ddos_trafic = 0

            for i in y_flow_pred:
                if i == 0:
                    legitimate_trafic = legitimate_trafic + 1
                else:
                    ddos_trafic = ddos_trafic + 1
                    victim = int(predict_flow_dataset.iloc[i, 5])%20




            self.logger.info("------------------------------------------------
-----------------------------")
            if (legitimate_trafic/len(y_flow_pred)*100) > 80:
                self.logger.info("legitimate trafic ...")
            else:
                self.logger.info("ddos trafic ...")
                self.logger.info("victim is host: h{}".format(victim))

            self.logger.info("------------------------------------------------
-----------------------------")

            file0 = open("PredictFlowStatsfile.csv","w")


file0.write('timestamp,datapath_id,flow_id,ip_src,tp_src,ip_dst,tp_dst,ip_proto,
icmp_code,icmp_type,flow_duration_sec,flow_duration_nsec,idle_timeout,hard_timeo
ut,flags,packet_count,byte_count,packet_count_per_second,packet_count_per_nsecon
d,byte_count_per_second,byte_count_per_nsecond\n')
            file0.close()

        except:
            pass
```

KGTB_2025

Файл collect_ddos_trafic.py

```
import switch
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub

from datetime import datetime

class CollectTrainingStatsApp(switch.SimpleSwitch13):
    def __init__(self, *args, **kwargs):
        super(CollectTrainingStatsApp, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self.monitor)

    #Asynchronous message
    @set_ev_cls(ofp_event.EventOFPStateChange,[MAIN_DISPATCHER,
DEAD_DISPATCHER])
    def state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                self.datapaths[datapath.id] = datapath

        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                del self.datapaths[datapath.id]


    def monitor(self):
        while True:
            for dp in self.datapaths.values():
                self.request_stats(dp)
            hub.sleep(10)


    def request_stats(self, datapath):
        self.logger.debug('send stats request: %016x', datapath.id)

        parser = datapath.ofproto_parser

        req = parser.OFPFlowStatsRequest(datapath)
        datapath.send_msg(req)

    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _flow_stats_reply_handler(self, ev):

        timestamp = datetime.now()
        timestamp = timestamp.timestamp()
        icmp_code = -1
        icmp_type = -1
        tp_src = 0
        tp_dst = 0
```

КбПЗ_2025

```
        file0 = open("FlowStatsfile.csv","a+")

        body = ev.msg.body
        for stat in sorted([flow for flow in body if (flow.priority == 1) ],
key=lambda flow:

(flow.match['eth_type'],flow.match['ipv4_src'],flow.match['ipv4_dst'],flow.match
['ip_proto'])):

            ip_src = stat.match['ipv4_src']
            ip_dst = stat.match['ipv4_dst']
            ip_proto = stat.match['ip_proto']

            if stat.match['ip_proto'] == 1:
                icmp_code = stat.match['icmpv4_code']
                icmp_type = stat.match['icmpv4_type']

            elif stat.match['ip_proto'] == 6:
                tp_src = stat.match['tcp_src']
                tp_dst = stat.match['tcp_dst']

            elif stat.match['ip_proto'] == 17:
                tp_src = stat.match['udp_src']
                tp_dst = stat.match['udp_dst']

            flow_id = str(ip_src) + str(tp_src) + str(ip_dst) + str(tp_dst) +
str(ip_proto)

            try:
                packet_count_per_second = stat.packet_count/stat.duration_sec
                packet_count_per_nsecond = stat.packet_count/stat.duration_nsec
            except:
                packet_count_per_second = 0
                packet_count_per_nsecond = 0

            try:
                byte_count_per_second = stat.byte_count/stat.duration_sec
                byte_count_per_nsecond = stat.byte_count/stat.duration_nsec
            except:
                byte_count_per_second = 0
                byte_count_per_nsecond = 0



file0.write("{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}\n
"
                .format(timestamp, ev.msg.datapath.id, flow_id, ip_src,
tp_src,ip_dst, tp_dst,
                        stat.match['ip_proto'],icmp_code,icmp_type,
                        stat.duration_sec, stat.duration_nsec,
                        stat.idle_timeout, stat.hard_timeout,
                        stat.flags, stat.packet_count,stat.byte_count,
                        packet_count_per_second,packet_count_per_nsecond,
                        byte_count_per_second,byte_count_per_nsecond,1))
        file0.close()
```

КФТЗ 2025

Файл switch.py

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ether_types

from ryu.lib.packet import in_proto
from ryu.lib.packet import ipv4
from ryu.lib.packet import icmp
from ryu.lib.packet import tcp
from ryu.lib.packet import udp

class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                          ofproto.OFPCML_NO_BUFFER)]
        self.add_flow(datapath, 0, match, actions)

    def add_flow(self, datapath, priority, match, actions, buffer_id=None,
idle=0, hard=0):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

        inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                             actions)]
        if buffer_id:
            mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                    idle_timeout=idle, hard_timeout=hard,
                                    priority=priority, match=match,
                                    instructions=inst)
        else:
            mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                    idle_timeout=idle, hard_timeout=hard,
                                    match=match, instructions=inst)

        datapath.send_msg(mod)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        # If you hit this you might want to increase
```

КБПЗ_2025

```
        # the "miss_send_length" of your switch
        if ev.msg.msg_len < ev.msg.total_len:
            self.logger.debug("packet truncated: only %s of %s bytes",
                              ev.msg.msg_len, ev.msg.total_len)
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        in_port = msg.match['in_port']

        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocols(ethernet.ethernet)[0]

        if eth.ethertype == ether_types.ETH_TYPE_LLDP:
            return
        dst = eth.dst
        src = eth.src

        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})

        # learn a mac address to avoid FLOOD next time.
        self.mac_to_port[dpid][src] = in_port

        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD

        actions = [parser.OFPActionOutput(out_port)]

        # install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:

            # check IP Protocol and create a match for IP
            if eth.ethertype == ether_types.ETH_TYPE_IP:
                ip = pkt.get_protocol(ipv4.ipv4)
                srcip = ip.src
                dstip = ip.dst
                protocol = ip.proto

                # if ICMP Protocol
                if protocol == in_proto.IPPROTO_ICMP:
                    t = pkt.get_protocol(icmp.icmp)
                    match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP,
                                            ipv4_src=srcip, ipv4_dst=dstip,
ip_proto=protocol,icmpv4_code=t.code,
                                            icmpv4_type=t.type)

                #  if TCP Protocol
                elif protocol == in_proto.IPPROTO_TCP:
                    t = pkt.get_protocol(tcp.tcp)
                    match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP,
                                            ipv4_src=srcip, ipv4_dst=dstip,
                                            ip_proto=protocol,
```

КСТЗ_2025

```
                                        tcp_src=t.src_port,
tcp_dst=t.dst_port,)

                #  If UDP Protocol
                elif protocol == in_proto.IPPROTO_UDP:
                    u = pkt.get_protocol(udp.udp)
                    match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP,
                                            ipv4_src=srcip, ipv4_dst=dstip,
                                            ip_proto=protocol,
                                            udp_src=u.src_port,
udp_dst=u.dst_port,)

                # verify if we have a valid buffer_id, if yes avoid to send both
                # flow_mod & packet_out
                if msg.buffer_id != ofproto.OFP_NO_BUFFER:
                    self.add_flow(datapath, 1, match, actions, msg.buffer_id,
idle=20, hard=100)
                    return
                else:
                    self.add_flow(datapath, 1, match, actions, idle=20,
hard=100)
        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data

        out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                                  in_port=in_port, actions=actions, data=data)
        datapath.send_msg(out)
```

КбПЗ_2025

Файл ML.py

```
from datetime import datetime

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

class MachineLearning():

    def __init__(self):

        print("Loading dataset ...")

        self.flow_dataset = pd.read_csv('FlowStatsfile.csv')

        self.flow_dataset.iloc[:, 2] = self.flow_dataset.iloc[:,
2].str.replace('.', '')
        self.flow_dataset.iloc[:, 3] = self.flow_dataset.iloc[:,
3].str.replace('.', '')
        self.flow_dataset.iloc[:, 5] = self.flow_dataset.iloc[:,
5].str.replace('.', '')

        self.X_flow = self.flow_dataset.iloc[:, :-1].values
        self.X_flow = self.X_flow.astype('float64')

        self.y_flow = self.flow_dataset.iloc[:, -1].values

        self.X_flow_train, self.X_flow_test, self.y_flow_train, self.y_flow_test
= train_test_split(self.X_flow, self.y_flow, test_size=0.25, random_state=0)

    def LR(self):

        print("----------------------------------------------------------------
-------------")
        print("Logistic Regression ...")

        self.classifier = LogisticRegression(solver='liblinear', random_state=0)
        self.Confusion_matrix()

    def KNN(self):

        print("----------------------------------------------------------------
-------------")
        print("K-NEAREST NEIGHBORS ...")

        self.classifier = KNeighborsClassifier(n_neighbors=5,
metric='minkowski', p=2)
        self.Confusion_matrix()
```

КБПЗ_2025

```
    def SVM(self):

        print("-------------------------------------------------------------------
-------------")
        print("SUPPORT-VECTOR MACHINE ...")

        self.classifier = SVC(kernel='rbf', random_state=0)
        self.Confusion_matrix()

    def NB(self):

        print("-------------------------------------------------------------------
-------------")
        print("NAIVE-BAYES ...")

        self.classifier = GaussianNB()
        self.Confusion_matrix()


    def DT(self):

        print("-------------------------------------------------------------------
-------------")
        print("DECISION TREE ...")

        self.classifier = DecisionTreeClassifier(criterion='entropy',
random_state=0)
        self.Confusion_matrix()

    def RF(self):

        print("-------------------------------------------------------------------
-------------")
        print("RANDOM FOREST ...")

        self.classifier = RandomForestClassifier(n_estimators=10,
criterion="entropy", random_state=0)
        self.Confusion_matrix()

    def Confusion_matrix(self):
        self.flow_model = self.classifier.fit(self.X_flow_train,
self.y_flow_train)

        self.y_flow_pred = self.flow_model.predict(self.X_flow_test)

        print("-------------------------------------------------------------------
-------------")

        print("confusion matrix")
        cm = confusion_matrix(self.y_flow_test, self.y_flow_pred)
        print(cm)

        acc = accuracy_score(self.y_flow_test, self.y_flow_pred)

        print("succes accuracy = {0:.2f} %".format(acc*100))
        fail = 1.0 - acc
```

КБПЗ_2025

```
        print("fail accuracy = {0:.2f} %".format(fail*100))
        print("----------------------------------------------------------------")

def main():

    start_script = datetime.now()

    ml = MachineLearning()

    start = datetime.now()
    ml.LR()
    end = datetime.now()
    print("LEARNING and PREDICTING Time: ", (end-start))

    start = datetime.now()
    ml.KNN()
    end = datetime.now()
    print("LEARNING and PREDICTING Time: ", (end-start))

     start = datetime.now()
     ml.SVM()
     end = datetime.now()
     print("LEARNING and PREDICTING Time: ", (end-start))

    start = datetime.now()
    ml.NB()
    end = datetime.now()
    print("LEARNING and PREDICTING Time: ", (end-start))

    start = datetime.now()
    ml.DT()
    end = datetime.now()
    print("LEARNING and PREDICTING Time: ", (end-start))

    start = datetime.now()
    ml.RF()
    end = datetime.now()
    print("LEARNING and PREDICTING Time: ", (end-start))

    end_script = datetime.now()
    print("Script Time: ", (end_script-start_script))

if __name__ == "__main__":
    main()
```

КПЗ_2025