Тернопільський національний економічний університет

На правах рукопису

КОМАР МИРОСЛАВ ПЕТРОВИЧ

УДК 004.056.53 : 004.492.3

ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ

05.13.06 – Інформаційні технології

Дисертація на здобуття наукового ступеня
кандидата технічних наук

Наукові керівники:
Саченко Анатолій Олексійович
доктор технічних наук, професор
Головко Володимир Адамович
доктор технічних наук, професор

Тернопіль – 2012

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 5
ВСТУП 6
РОЗДІЛ 1 АНАЛІЗ ВІДОМИХ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ 14
1.1 Стан розробки технологій виявлення і класифікації атак на інформаційні телекомунікаційні мережі 14
1.2 Огляд методів штучного інтелекту, на яких базуються інформаційні технології виявлення і класифікації атак 20
1.3 Аналіз наборів тестових даних і оцінка достовірності інформаційних технологій 28
1.4 Напрями дослідження і постановка задачі 38
РОЗДІЛ 2 НЕЙРОМЕРЕЖЕВІ МЕТОДИ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ 41
2.1 Узагальнена функціональна модель прийняття рішень при виявленні і класифікації атак 41
2.2 Вибір базової архітектури нейромережевого детектора 44
2.3 Метод побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі 52
2.3.1 Структура нейромережевого детектора 52
2.3.2 Навчання нейромережевого детектора 56
2.3.3 Структура нейронів прихованого шару і навчальної вибірки 59
2.4 Метод побудови сукупного класифікатора для ієрархічної класифікації атак 61
2.4.1 Стиснення вхідних даних на основі методу головних компонент .. 61
2.4.2 Сукупний класифікатор для ієрархічної класифікації атак 69
Висновки до розділу 2 74

РОЗДІЛ 3 КОМБІНОВАНИЙ МЕТОД ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ 77
3.1 Основи технології виявлення і класифікації атак з використанням імунних систем 77
3.2 Розробка комбінованого методу виявлення і класифікації атак на інформаційні телекомунікаційні мережі 79
3.3 Навчання і функціонування нейромережевих імунних детекторів 85
3.4 Експериментальні дослідження комбінованого методу виявлення і класифікації атак 88
Висновки до розділу 3 96
РОЗДІЛ 4 ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ 98
4.1 Структура програмного забезпечення та алгоритми функціонування нейромережевої імунної системи виявлення і класифікації атак 98
4.2 Реалізація програмного забезпечення 110
4.3 Статистична оцінка достовірності розробленої інтелектуальної інформаційної технології 114
Висновки до розділу 4 128
ВИСНОВКИ 130
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 133
Додаток А Типи мережевих атак 155
Додаток Б Атрибути мережевого трафіку 158
Додаток В Адаптація параметрів з’єднань 161
Додаток Г Приклади векторів для навчання НМ 164
Додаток Д Програмна реалізація нейронної мережі MPL 173
Додаток Е Програмна реалізація нейронної мережі RBF 175
Додаток Ж Програмна реалізація нейронної мережі LVQ 177
Додаток К Розподіл в тривимірному просторі мережевих атак і нормальних з’єднань 179

Додаток Л Фрагменти лістингів програмного забезпечення системи виявлення і класифікації атак 190
Додаток М Акти впровадження результатів дисертаційного дослідження .. 198

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DOS – Denial of Service

FPR – False Positives Rate

IDS – Intrusion Detection System

PCA – Principal Component Analysis

R2L – Remote-to-local

ROC – Receiver Operator Characteristic

TPR – True Positives Rate

U2R – User-to-root

АІС – Автоматизована інформаційна система

БІС – Біологічна імунна система

ІТ – Інформаційна технологія

ІТМ – Інформаційна телекомунікаційна мережа

ІІТ – Інтелектуальна інформаційна технологія

КС – Комп’ютерна система

ЛОМ – Локальна обчислювальна мережа

НД – Нейромережевий детектор

НІД – Нейромережевий імунний детектор

НМ – Нейронна мережа

НСД – Несанкціонований доступ

ОС – Операційна система

ПЗ – Програмне забезпечення

СВА – Система виявлення атак

ШІС – Штучна імунна система

ШНМ – Штучна нейронна мережа

ВСТУП

Актуальність теми. Останнім часом відбулося злиття комп'ютерних мереж, інформаційних і телекомунікаційних технологій, що дозволило утворити сучасні інформаційні телекомунікаційні мережі (ІТМ), які є складною розподіленою системою, що характеризується наявністю множини взаємодіючих ресурсів, системних та прикладних інформаційних і телекомунікаційних процесів. У таких умовах важливою науково-технічною задачею є забезпечення цілісності, достовірності та конфіденційності інформації. Інформаційні телекомунікаційні мережі піддаються різного роду загрозам, і користувач не може бути впевнений у захищеності важливої інформації, оскільки кіберзлочинці продовжують удосконалювати і розробляти методи і засоби організації мережевих атак. Наприклад, за даними «Лабораторії Касперського» щодня в світі з'являється близько 70 тисяч нових шкідливих програм, близько 200 мільйонів мережевих атак блокується щомісячно [1], а в 2011 році система IDS Kaspersky Internet Security відбила майже 3 мільярди мережевих атак [2].

Аналіз найбільш поширених інформаційних технологій виявлення мережевих атак, таких як сигнатурний і статистичний аналіз, показує їх нездатність виявляти нові або невідомі атаки, для яких характерна відсутність записів в системі про них. Інформаційні технології, які базуються на евристичних методах, мають високу ймовірність помилкових спрацювань. Ситуація, що склалася, стимулює пошук і розробку нових методів та інформаційних технологій, спрямованих на підвищення достовірності виявлення і класифікації атак на ІТМ.

Одним з перспективних напрямків є застосування методів штучного інтелекту. Дослідженнями в цій сфері займаються Головко В.А. [3–12], Городецький В.І. [13–17], Котенко І.В. [18–21], Широчин В.П. [22–30], Cannady J. [31], Dasgupta D. [32], De Castro L. [33–35], Grediaga A. [36], Ibrahim L.M. [37], Jing Xiao-Pei [38] Moradi M. [39], Mukkamala S. [40], Murad

Abdo Rassam [41], Novosad T. [42], Ramadas M. [43], Sammany M. [44], Sridevi R. [45] та ін. Разом з тим, дані підходи характеризуються наявністю ряду вузьких місць, таких як складність створення або вибору необхідних детекторів атак, складність адаптації до невідомих атак, здатність коректно працювати тільки на невеликих наборах даних.

Для усунення цих недоліків запропоновано використати інтеграцію методів штучних нейронних мереж і штучних імунних систем, оскільки вони поєднують високу достовірність виявлення і класифікації атак з властивістю до навчання, адаптації, самоорганізації і пам'яті.

Враховуючи вищесказане, можна зробити висновок про актуальність досліджень, спрямованих на розробку інформаційної технології на базі теорії штучних нейронних мереж та штучних імунних систем з метою підвищення достовірності виявлення та класифікації атак на ІТМ.

Зв'язок роботи з науковими програмами, планами, темами. Основний зміст дисертаційної роботи складають теоретичні і практичні результати досліджень, проведених автором при виконанні держбюджетної науково-дослідної роботи Тернопільського національного економічного університету «Методи та засоби виявлення вторгнень на комп'ютерні системи» (номер державної реєстрації – 0110U000786, термін виконання 06.2010 р. – 06.2012 р.), де автор був відповідальним виконавцем.

Мета і задачі дослідження. Метою дисертаційної роботи є створення нової інтелектуальної інформаційної технології на базі теорії штучних нейронних мереж і штучних імунних систем для підвищення достовірності виявлення і класифікації атак на ІТМ.

Для досягнення поставленої мети вирішуються наступні задачі:

1. Аналіз існуючих інформаційних технологій для визначення шляхів підвищення достовірності виявлення та класифікації атак на інформаційні телекомунікаційні мережі.

2. Розробка узагальненої функціональної моделі прийняття рішень при виявленні і класифікації атак на інформаційні телекомунікаційні мережі.

3. Вибір нейронної мережі для виявлення і класифікації атак на інформаційні телекомунікаційні мережі.

4. Розробка методу побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі на базі нейронної мережі, що характеризується малим обсягом навчальної вибірки.

5. Розробка методу побудови сукупного класифікатора для ієрархічної класифікації атак на інформаційні телекомунікаційні мережі на основі багатоканальних нейромережевих детекторів.

6. Розробка комбінованого методу виявлення і класифікації атак на інформаційні телекомунікаційні мережі на основі інтеграції нейромережевих детекторів в штучну імунну систему.

7. Створення інтелектуальної інформаційної технології для вирішення задачі виявлення та класифікації атак на інформаційні телекомунікаційні мережі на основі запропонованих методів.

8. Проведення статистичної оцінки достовірності розробленої інформаційної технології з метою порівняння з відомими рішеннями.

Об'єкт дослідження – процес обробки інформації при виявленні і класифікації атак на інформаційні телекомунікаційні мережі.

Предмет дослідження – методи і засоби інтелектуальної інформаційної технології виявлення і класифікації атак на ІТМ.

Методи дослідження. Для вирішення поставлених задач використовуються методи системного аналізу, математичної статистики, теорії розпізнавання образів, теорії штучних нейронних мереж, нейрообчислень і штучних імунних систем.

Наукова новизна дослідження. В ході виконання дослідження були отримані наступні основні результати, що відображають наукову новизну роботи:

1. Вперше розроблено комбінований метод виявлення і класифікації атак на інформаційні телекомунікаційні мережі шляхом інтеграції нейромережевих детекторів в штучну імунну систему, що дозволило їм

адаптуватися до невідомих атак за рахунок здійснення операцій клонування і мутації.

2. Вдосконалено метод побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі, в якому, на відміну від відомих, нейронні елементи в прихованому шарі розділені на два класи, які характеризують атаку або нормальне з'єднання, що дало можливість окремо здійснити кластеризацію атак і нормальних з'єднань в прихованому шарі і підвищити достовірність виявлення атак при малому об'ємі навчальної вибірки.

3. Вдосконалено метод побудови сукупного класифікатора для ієрархічної класифікації атак на інформаційні телекомунікаційні мережі на основі багатоканальних нейромережевих детекторів, який, на відміну від відомих, поєднує використання методу головних компонент, об'єднання і усунення конфліктів між навченими на певний тип атак нейромережевими детекторами, що дозволило зменшити розмірність аналізованої інформації та класифікувати мережеві атаки.

4. Отримала подальший розвиток інтелектуальна інформаційна технологія виявлення і класифікації атак на інформаційні телекомунікаційні мережі на основі використання операцій навчання, відбору, клональної селекції, мутації та імунної пам'яті з метою формування якнайкращої популяції детекторів, яка, на відміну від відомих, характеризується генеруванням множини нейромережевих детекторів для кожного типу мережевої атаки, що дало можливість підвищити достовірність виявлення і класифікації як відомих, так і невідомих атак.

Обгрунтованість і достовірність наукових положень, висновків і рекомендацій. Всі наукові положення, висновки і рекомендації дисертаційної роботи забезпечуються коректністю постановки наукової задачі і виконання теоретичних досліджень з використанням математичного апарату, програмною реалізацією методів виявлення і класифікації атак на ІТМ, результатами експериментальних досліджень, ефективним практичним

впровадженням результатів дисертаційних досліджень, що продемонструвало відповідність теоретичних досліджень практичним результатам.

Практичне значення одержаних результатів. В результаті виконаного дисертаційного дослідження, на основі розроблених методів і засобів, реалізована та впроваджена нова нейромережева імунна система виявлення і класифікації як відомих, так і невідомих атак на ІТМ. Результати експериментальних досліджень з використанням розробленого програмного забезпечення підтверджують вірність наукових положень дисертаційної роботи, оскільки впровадження інтелектуальної інформаційної технології підвищує достовірність виявлення і класифікації атак на ІТМ на 0,27-8,5% у порівнянні з відомими рішеннями.

Теоретичні та практичні результати роботи використані:

– Приватним підприємством «МагнетікВан» (м. Тернопіль) (акт впровадження від 12.10.2012 р.);

– Товариством з обмеженою відповідальністю «СофтІнвест» (м. Брест, Республіка Білорусь) (акт впровадження від 25.09.2012 р.);

– Науково-дослідним інститутом інтелектуальних комп'ютерних систем (Тернопільський національний економічний університет, Інститут кібернетики ім. В.М. Глушкова НАН України) в рамках науково-дослідної роботи на тему: «Методи та засоби виявлення вторгнень на комп'ютерні системи» (акт впровадження від 18.10.2012 р.);

– в рамках двостороннього договору про партнерство, співпрацю і науковий обмін між Тернопільським національним економічним університетом і Брестським державним технічним університетом (акт впровадження від 19.09.2012 р.);

– у навчальному процесі Тернопільського національного економічного університету при викладанні дисциплін «Комп'ютерні мережі», «Телекомунікаційні системи», «Теорія нейронних мереж», «Методи і системи штучного інтелекту» (акт впровадження від 05.10.2012 р.).

Особистий внесок здобувача. Всі основні результати, що виносяться

на захист, отримані здобувачем особисто [46–57]. У роботах, опублікованих у співавторстві, здобувачеві належить: у [58] – структура штучної імунної системи виявлення і класифікації атак на інформаційні телекомунікаційні мережі, що дало можливість інтегрувати нейромережеві детектори в штучну імунну систему; у [59] – вдосконалення і експериментальні дослідження методів і алгоритмів виявлення мережевих атак шляхом застосування нейромережевих детекторів на основі штучних нейронних мереж, що дало можливість побудувати нову інтелектуальну інформаційну технологію виявлення і класифікації атак на ІТМ; у [60] – метод побудови нейромережевого детектора і експериментальні дослідження застосування методу головних компонент для підвищення достовірності виявлення атак, що дало можливість побудувати сукупний ієрархічний класифікатор; у [61] – експериментальні дослідження інтелектуальної системи виявлення шкідливих дій на комп'ютерні системи, що дозволило дослідити можливість нейромережевих детекторів виявляти невідомі вторгнення; у [62] – комбінована інформаційна технологія виявлення і класифікації мережевих атак на основі методу головних компонент і нейромережевого детектора і експериментальні дослідження, що дало можливіть підвищити достовірність процесу виявлення і класифікації атак на ІТМ; у [63] – метод виявлення і класифікації атак на інформаційні телекомунікаційні мережі, заснований на інтеграції нейромережевих детекторів в штучну імунну систему і оцінка достовірності запропонованого підходу на основі ROC-аналізу; у [64] – метод побудови сукупного класифікатора для ієрархічної класифікації атак на основі багатоканальних нейромережевих детекторів, який поєднує використання методу головних компонент, об'єднання і усунення конфліктів між нейромережевими детекторами; у [65] – спосіб виявлення комп'ютерних атак нейромережевою штучною імунною системою.

Апробація результатів дисертації. Основні положення і результати дисертаційної роботи були висвітлені та обговорені на міжнародних і національних конференціях, а саме:

– міжнародній конференції «Комп'ютерні системи і мережеві технології» (Київ, 2009);

– республіканській конференції молодих вчених і студентів «Сучасні проблеми математики і обчислювальної техніки» (Брест, Республіка Білорусь, 2009);

– міжнародній конференції «Modern Problems of Radio Engineering, Telecommunications and Computer Science» (Львів-Славське, 2010);

– міжнародній науково-технічній конференції «Internet–Education–Science» (Вінниця, 2010);

– міжнародній конференції молодих вчених і студентів «Актуальні задачі сучасних технологій» (Тернопіль, 2010);

– міжнародній конференції «Методи та засоби кодування, захисту й ущільнення інформації» (Вінниця, 2011);

– міжнародній конференції «Обчислювальний інтелект (результати, проблеми, перспективи)» (Черкаси, 2011);

– міжнародній конференції «Інформаційні технології та безпека в управлінні» (Севастополь, 2011);

– міжнародній конференції «Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications» (Прага, Чехія, 2011);

– загальноуніверситетській науковій конференції професорсько-викладацького складу, докторантів, аспірантів і здобувачів вченого ступеня ТНЕУ «Економічні, правові, інформаційні і гуманітарні проблеми розвитку України в умовах проведення системних реформ» (Тернопіль, 2012);

– міжнародній конференції «Захист інформації і безпека інформаційних систем» (Львів, 2012);

– міжнародній конференції «CAD in Machinery Design. Implementation and Educational Issues» (Львів, 2012).

Публікації. Основні результати дисертаційної роботи висвітлені у 20 друкованих працях, в т.ч. 6 статей (з них 5 одноосібні) у фахових виданнях України; один патент України; одна стаття у міжнародному науковому

журналі, одна стаття у вітчизняному науковому журналі, 11 робіт у збірниках міжнародних і всеукраїнських наукових конференцій.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. Загальний обсяг роботи становить 204 сторінки, з яких 132 сторінки основного тексту. Робота містить 47 рисунків, 36 таблиць і 10 додатків. Список використаних джерел включає 192 найменувань на 22 сторінках.

РОЗДІЛ 1
АНАЛІЗ ВІДОМИХ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ

1.1 Стан розробки технологій виявлення і класифікації атак на інформаційні телекомунікаційні мережі

Злиття комп'ютерних мереж, інформаційних та телекомунікаційних технологій дозволило утворити сучасні інформаційні телекомунікаційні мережі, які є складною розподіленою системою, що характеризується наявністю множини взаємодіючих ресурсів і системних та прикладних інформаційних і телекомунікаційних процесів, які одночасно протікають в системі.

Інформаційна телекомунікаційна мережа (рисунок 1.1) [66], в загальному випадку, включає наступні компоненти:

– мережа доступу (access network) – для концентрації інформаційних потоків, що надходять по каналах зв'язку від обладнання користувачів, в порівняно невеликій кількості вузлів магістральної мережі;

– магістраль (backbone або core network) – об'єднує окремі мережі доступу, забезпечуючи транзит трафіку між ними по високошвидкісних каналах;

– інформаційні центри або центри управління сервісами (data centers або services control point) – це власні інформаційні ресурси мережі, на основі яких здійснюється обслуговування користувачів.

У таких умовах важливою науково-технічною задачею є забезпечення цілісності, достовірності та конфіденційності інформації, що передається по каналах передачі ІТМ.

На сьогоднішній день склалася така ситуація, коли пропоновані методи захисту ІТМ не здатні на належному рівні забезпечити інформаційну безпеку. ІТМ безперервно піддаються різного роду загрозам і користувач не може

бути впевнений в захищеності важливої інформації, оскільки кіберзлочинці продовжують удосконалювати і розробляти методи та засоби організації мережевих атак.

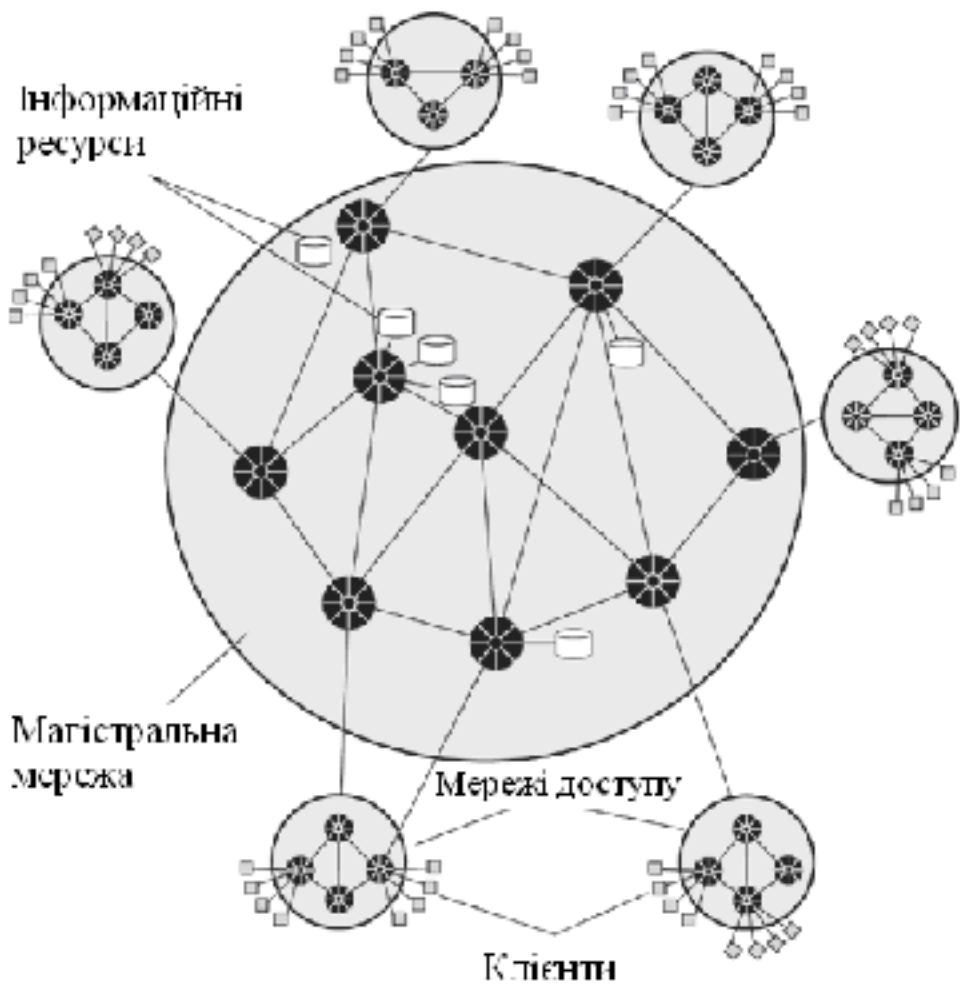


Рисунок 1.1 – Загальна схема інформаційної телекомунікаційної мережі

Розглянемо визначення в даній предметній області.

Визначення 1.1. Телекомунікаційна мережа – комплекс технічних засобів телекомунікацій та споруд, призначених для маршрутизації, комутації, передавання та/або приймання знаків, сигналів, письмового тексту, зображень та звуків або повідомлень будь-якого роду по радіо, проводових, оптичних чи інших електромагнітних системах між кінцевим обладнанням [67].

Визначення 1.2. Інформаційна безпека телекомунікаційних мереж – властивість телекомунікаційних мереж забезпечувати захист від знищення, перекручення, блокування інформації, її несанкціонованого витоку або від порушення встановленого порядку її маршрутизації [67].

Визначення 1.3. Загроза – будь-які обставини або події, які можуть бути причиною порушення політики безпеки інформації і/або нанесення

збитків автоматизованій системі [68, 69].

Визначення 1.4. Атака – спроба реалізації загрози [68].

До цих пір немає точного визначення терміну «атака» (вторгнення, напад). Кожен фахівець в області безпеки трактує його по-своєму.

Визначення 1.5. Атака на інформаційну систему – навмисні дії зловмисника, що використовують вразливості інформаційної системи, які приводять до порушення доступності, цілісності і конфіденційності інформації, що обробляється [70].

Визначення 1.6. Мережева атака – дія, що здійснюється зловмисником і направлена на реалізацію загрози. Дія полягає в пошуку будь-якої вразливості комп'ютерної системи із використанням як спеціалізованих програмних засобів, так і за допомогою різних психологічних прийомів [71].

Визначення 1.7. Віддалена мережева атака – це інформаційна руйнуюча дія на розподілену обчислювальну систему, яка здійснюється програмними методами за допомогою каналів зв'язку [72].

Визначення 1.8. Інформаційна технологія – технологічний процес, предметом переробки і результатом якого є інформація [73].

Визначення 1.9. Інтелектуальна інформаційна технологія – прийоми, способи і методи виконання функцій збору, зберігання, обробки, передачі і використання знань [74].

Існують різні типи класифікації атак [75–79].

Залежно від техніки, що використовується при здійсненні несанкціонованих дій на комп'ютерну систему, виділяють чотири основні класи мережевих атак (denial of service, user-to-root, remote-to-local, probe), кожен з яких складається з декількох типів [80, 81]. Розглянемо кожен з класів атак докладніше.

DOS (denial of service, відмова в обслуговуванні) атаки. Це мережеві атаки, направлені на виникнення ситуації, коли в системі, що атакується, відбувається відмова в обслуговуванні. Дані атаки характеризуються генерацією великого об'єму трафіку, що приводить до перевантаження і

блокування сервера. Виділяють шість типів *DOS*-атак: *back, land, neptune, pod, smurf, teardrop* [81]:

U2R (user-to-root) атаки. Атаки даного класу передбачають отримання зареєстрованими користувачами привілеїв локального суперкористувача (адміністратора). Виділяють чотири типи *U2R*-атак: *buffer_overflow, loadmodule, perl, rootkit* [81]:

R2L (remote-to-local) атаки. Такі атаки характеризуються отриманням доступу незареєстрованого користувача до комп'ютера з боку віддаленої машини. Виділяють вісім типів *R2L*-атак: *ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster* [81]:

Probe-атаки грунтуються на процесі сканування мережевих портів віддаленої машини з метою отримання конфіденційної інформації. Виділяють чотири типи *Probe*-атак: *ipsweep, nmap, portsweep, satan* [81]:

Детальний опис типів атак представлений в додатку А.

Виявлення і класифікація мережевих атак на комп'ютерну систему відбувається за допомогою аналізу інформації в каналах обміну даними інформаційних телекомунікаційних мереж. Для розуміння даного процесу розглянемо параметри мережевого з'єднання, які аналізуються для забезпечення безпеки комп'ютерних систем.

Дані в інформаційних телекомунікаційних мережах передаються у вигляді мережевих пакетів. У структурі мережевого пакету виділяють три основні поля (рисунок 1.2): заголовок пакету, поле даних пакету, кінцевик пакету.

Заголовок пакету містить стартову комбінацію, яка забезпечує налаштування мережевого обладнання на прийом і обробку пакету, а також мережеві адреси приймача і передавача пакету та деяку загальну службову інформацію.

Поле даних пакету містить в собі власне інформацію, яка і передається від передавача до приймача.

Кінцевик пакету містить в собі контрольну суму, яка дозволяє

приймати рішення про успішність передачі інформації, стопову комбінацію, яка служить для інформування про закінчення пакету, а також деяку службову інформацію.

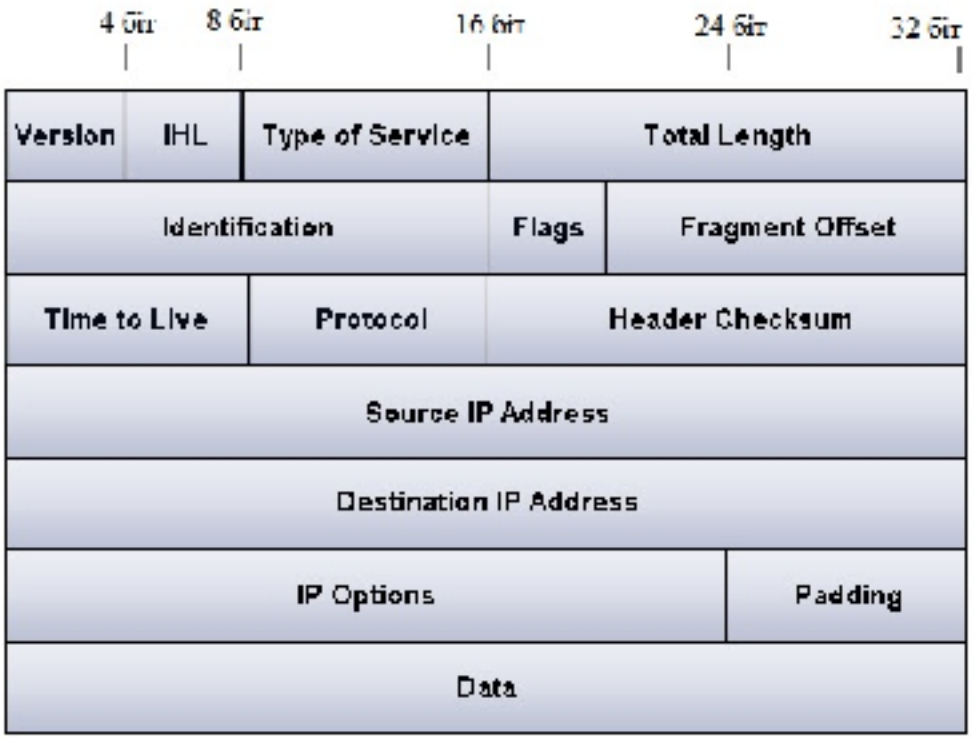


Рисунок 1.2 – Структура мережевого пакету

Виділяють 41 параметр мережевого з'єднання, які, у свою чергу, об'єднані у три групи [81]:

а) вбудовані параметри;

б) параметри контенту;

в) параметри трафіку.

Розглянемо існуючі технології виявлення і класифікації атак на інформаційні телекомунікаційні мережі.

У найпростішому випадку система захисту від мережевих атак може бути міжмережевим екраном (брандмауер, firewall) [70, 82].

Міжмережевий екран – це програмний або апаратний засіб фільтрації мережевого трафіку за допомогою аналізу його параметрів, таких як адреси джерела і приймача, типів мережевих протоколів і служб і т. д.

Контроль і фільтрацію трафіку міжмережевий екран здійснює у відповідності до заданих правил. Правила можуть бути жорстко задані розробником конкретного мережевого екрану або можуть змінюватися в процесі функціонування і взаємодії з користувачем.

Головною відмінністю мережевого екрану від системи виявлення вторгнень (див. нижче) є те, що в ньому відсутній аналіз вмісту пакетів, що передаються.

Міжмережеві екрани виконують набір функцій по захисту віддаленої системи від мережевих атак, що включають фільтрацію віддаленого доступу до незахищених служб, протидію отриманню доступу до закритої інформації, контроль доступу до вузлів обчислювальної мережі, реєстрацію в системних журналах здійснених спроб доступу до системи, регламентацію порядку доступу до мережі або системи, повідомлення користувача про підозрілу діяльність, що включає зондування мережі, мережеві атаки і т. д. На підставі аналізу активної мережевої діяльності з метою здійснення інформаційної безпеки екрани можуть блокувати окремі мережеві служби.

Серед переваг міжмережевих екранів можна виділити високу швидкість обробки вхідних і вихідних мережевих пакетів.

Недоліками ж є: низький рівень захисту, оскільки відсутній аналіз вмісту пакетів; відсутність захисту від внутрішніх загроз; відсутність захисту від завантаження шкідливих програм; обмеження доступу до потрібних сервісів.

Система виявлення вторгнень (Intrusion Detection System - IDS) [80, 85, 85] на сьогоднішній день є невід'ємною частиною системи безпеки будь-якої комп'ютерної системи, підключеної до локальної або глобальної комп'ютерної мережі. IDS, як правило, це програмний або апаратний засіб, що є «фільтром» і знаходиться між комп'ютерною системою і комп'ютерною мережею. IDS здатна виявляти деякі типи шкідливої активності, що включає: мережеві атаки проти вразливих сервісів; неавторизований доступ до важливих файлів; активність деяких шкідливих програм. Система аналізує параметри вхідного і вихідного трафіку з метою виявлення фактів неавторизованого доступу. IDS перехоплює весь мережевий трафік і аналізує вміст кожного пакету на наявність шкідливих компонентів. Класифікація систем виявлення атак представлена в [80, 86–89].

Системи виявлення вторгнень мають очевидні переваги перед міжмережевими екранами. Проте, окрім переваг, існуючі системи мають ряд істотних недоліків [70].

1.2 Огляд методів штучного інтелекту, на яких базуються інформаційні технології виявлення і класифікації атак

За час існування проблеми виявлення мережевих атак, розроблено багато методів, на яких базуються інформаційні технології виявлення і класифікації атак на ІТМ.

Проведений аналіз найбільш поширених інформаційних технологій виявлення і класифікації мережевих атак. Сигнатурний і статистичний аналіз показав нездатність виявляти нові або невідомі атаки, які характеризуються відсутністю записів в системі про них. У свою чергу, інформаційні технології, які базуються на евристичних методах, мають високу ймовірність помилкових спрацювань. Така ситуація стимулює пошук і розробку нових методів та інформаційних технологій, спрямованих на підвищення достовірності виявлення і класифікації атак на ІТМ.

Окрім зазначених недоліків в [90], такі методи, як аналіз систем станів, графи сценаріїв атак, експертні системи, методи, засновані на специфікаціях, і сигнатурні методи, не є адаптивними, а значить, не підходять для виявлення нових атак і нових видів вже існуючих атак.

Завдяки розвитку методів штучного інтелекту, з середини 90-х років відбувається перехід з ручних методів аналізу поведінки складних систем на автоматичні методи аналізу. Методи штучного інтелекту відомі, перш за все, такими характеристиками, як здатність до адаптації і навчання на помилках, високі обчислювальні швидкості, робота із зашумленими даними. Найбільш використовуваним підходом в області аналізу мережевого трафіку з метою виявлення аномалій є: штучні нейронні мережі, нечіткі системи, методи еволюційного програмування, штучні імунні системи. Розглянемо

детальніше відомі роботи по аналізу мережевого трафіку з метою виявлення в ньому аномалій.

Штучні нейронні мережі (ШНМ) складаються з набору зв'язаних один з одним обчислювальних елементів, так званих нейронів. Такі мережі характеризуються здатністю до навчання на прикладах і узагальнення результатів навчання, і здатні обробляти зашумлені і неповні дані, внаслідок чого отримали широке застосування для вирішення задач виявлення і класифікації мережевих атак [90].

IDS на основі нейронних мереж можна розділити на чотири категорії [92]. Перша категорія – IDS на основі багатошарових мереж прямого розповсюдження (Multi-Layer Feed Forward Neural Network – MLFF), таких як багатошаровий персептрон (Multi-Layer Perceptron – MLP) і нейронна мережа зворотного розповсюдження (Back Propagation – BP).

Так, Cannady в [31] застосував багатошаровий персептрон для аналізу мережевого трафіку з метою виявлення зовнішніх атак. Передбачалося, що застосування нейронної мережі дозволить уникнути більшості труднощів, пов'язаних із застосуванням методів виявлення атак, заснованих на правилах. Нейронна мережа складалася з девять вхідних і двох вихідних нейронів. Для навчання нейронної мережі використовувалася навчальна вибірка розмірністю 10000 мережевих пакетів, а час навчання становив 26 годин. До недоліків застосування MLP у такому вигляді можна віднести нездатність виявлення комбінованих атак, а тільки поодиноких. Також, достовірність виявлення безпосередньо залежала від якості навчальної вибірки.

У роботах [31–40] використовуються нейронні мережі MLFF для виявлення аномалій на основі аналізу поведінки користувача. Інші дослідники, наприклад, в [95–39], використовували MLP для виявлення аномалій.

У роботах [96, 97] використовується нейронна мережа з тимчасовою затримкою (Time Delay Neural Network – TDNN).

Інші дослідники порівнювали ефективність MLFF з іншими

нейронними мережами, наприклад, Siddiqui [98] у 2004 році порівнював ефективність BP з нечіткою ARTMAP, Grediaga [36] у 2006 році порівнював ефективність MLFF з картами Кохонена (SOM), що самоорганізовуються, Чжан [99] у 2004 році зробив порівняння між MPL і RBF мережами в IDS. Як було показано, нейронні мережі MLFF мають нижчу достовірність виявлення, ніж SOM.

До другої категорії відносяться IDS, які засновані на нейронних мережах Cerebellar Model Articulation Controller (CMAC) і Елмана. Cannady у в 2000 році запропонував використовувати CMAC, які мають широкі можливості для он-лайн навчання і адаптації, завдяки чому така система проводить виявлення атак, на яких навчена, а також автономне посилення і самонавчання до нових атак, зокрема завдяки зворотному зв'язку з системою, що захищається [100]. Debar та ін. [101] у 1999 році використовували спрощену рекурентну мережу ЕЛЬМАН і рекурентну багатошарову мережу зворотного розповсюдження, щоб прогнозувати команди. У [102] описано застосування моделі детектування аномалій в мережевому трафіку на основі використання рекурентної нейронної мережі Елмана, як класифікатора мережевих атак. Дані для навчання формувалися так, щоб аналіз мережевого трафіку відбувався не тільки на основі метаданих, взятих із заголовків пакетів, але також і з поля даних пакетів. Такий підхід дозволив покращити достовірність виявлення невідомих мережевих атак.

Третя категорія – системи IDS, які засновані на неконтрольованих нейронних мережах з використанням неконтрольованого навчання нейронних мереж для класифікації і візуалізації вхідних даних системи, щоб розпізнати нормальну поведінку і відрізнити від ненормальної. Більшість систем в цій категорії використовують карти Кохонена, що самоорганізовуються (Self-organizing map – SOM). Fox був першим, хто застосував SOM, щоб дізнатися характеристики нормальної поведінки системи і виявляти статистичні відхилення [103]. Rhodes та ін. [104], Höglund та ін. [105], Lichodzijewski та ін. [106] і Ramadas [43] навчали SOM на

зібраних нормальних даних з UNIX аудиту і використовували її для виявлення аномальної активності користувачів.

У [104] після детального вивчення мережевих пакетів виявили, що кожен окремий протокол має унікальну структуру і функціональність. В результаті, для аналізу мережевого трафіку було запропоновано використовувати багатошарову SOM, в якому кожен рівень відповідає певному протоколу. Передбачалося, що після навчання характеристик нормального трафіку, мережа коректно класифікуватиме аномальну поведінку в мережі. Карти Кохонена, що самоорганізовуються, автоматично розподіляють вхідні дані під час навчання на категорії і надалі показують, як нові дані співвідносяться з тією або іншою категорією. У результаті, розроблена система змогла виявити дві змодельовані атаки класу buffer_overflow на сервер.

До подібного висновку прийшли і автори [107]. Вони згрупували 41 параметр мережевого з'єднання в три групи і навчали тришарові SOM на виділених параметрах, що в результаті дозволило істотно скоротити рівень помилок першого роду.

Четверта категорія – системи IDS, які засновані на гібридних нейронних мережах. Jirapummin [108] запропонував використовувати гібридну нейронну мережу, для виявлення вторгнень – SOM, для класифікації вторгнень – мережа «пружиного розповсюдження» (resilient propagation – RPROP). Horeis [109] використовував комбінацію SOM і радіально-базисних нейронних мереж (Radial Basis Function – RBF). Система дає в цілому кращі результати, ніж виявлення вторгнень тільки на основі мереж RBF.

Застосування методів нечіткої логіки у вирішенні задач виявлення вторгнень доцільне відповідно до двох основних причин. Перша причина – мережевий трафік має множину чисельних атрибутів, які необхідно аналізувати при вирішенні проблеми захисту. Друга – безпека сама має на увазі нечіткість, оскільки різниця між нормальною поведінкою і атакою може

бути вельми незначною і тому недостатньо детермінованою.

Моделі виявлення вторгнень, побудовані із застосуванням методів нечіткої логіки, базуються на використанні нечітких правил або нечітких класифікаторів [110]. Так в [111] Dickerson та ін. запропонували механізм FIRE (Fuzzy Intrusion Recognition Engine) для виявлення шкідливої активності в мережі. Тут порції даних класифікуються на основі статичних метрик, що дозволяє надалі створити правила нечіткої логіки для класифікації вхідних мережевих даних. Основним недоліком запропонованої системи є те, що правила створюються вручну, а не виробляються автоматично. Процес генерації правил є трудомістким і накладає серйозні обмеження на розвиток системи.

У ряді інших робіт для виявлення вторгнень використовується методи еволюційних обчислень. Так в [112] і [113] застосований генетичний алгоритм. Вирішення проблеми кодується за допомогою бінарного рядка, де довжина рядка є кількістю атак, а значення «1» або «0» в рядку позначають наявність атаки.

Штучні імунні системи (ШІС) – порівняно молодий напрям в області штучного інтелекту, знайшли широке застосування в області захисту інформації.

Ідея створення штучних імунних систем з'явилася в результаті вивчення процесів біологічного імунітету, який захищає організм від хвороботворних бактерій і вірусів, виявляючи і знищуючи їх [114–120].

В результаті проведеного аналізу біологічної імунної системи був зроблений висновок, що дана система є надійним механізмом виявлення аномалій у вигляді хвороботворних бактерій і вірусів. Така система характеризується здатністю до класифікації об'єктів різного класу, а також наявністю механізмів боротьби з виявленими інфекціями.

Завдяки своїм особливостям і характеристикам, імунна система представляє великий інтерес в області обробки масивів даних і захисту інформації.

Сфера застосування на сьогоднішній день включає наступні області (але не обмежується ними): методи обчислень, когнітивні моделі, виявлення аномалій і несправностей, багатоагентні системи, моделі самоорганізації, моделі колективного інтелекту, системи пошуку і оптимізації, моделі автономних розподілених систем, моделі штучного життя, системи комп'ютерної безпеки, методи витягування інформації, обробка сигналів і зображень та ін. [114, 121–131].

Зазначені характеристики і можливості доводять перспективність використання основних концепцій імунітету у вирішенні складних комп'ютерних задач, у тому числі і задач виявлення та класифікації мережевих атак.

Системи виявлення вторгнень, що базуються на ШІС, здатні виявляти аномалії в масивах даних, у тому числі і наявність мережевих атак в мережевому трафіку. Перевагою класичних моделей виявлення вторгнень, що базуються на методах ШІС, є те, що вони здатні генерувати аномальні патерни, грунтуючись на нормальному трафіку. Таким чином, для навчання таких систем досить мати тільки дані про нормальний мережевий трафік, а система, навчившись, розпізнаватиме будь-яку аномальну поведінку в мережі.

У роботах [132–45] представлені підходи до виявлення вторгнень на базі методів ШІС. Так, наприклад, в роботі [38] використовується генетичний алгоритм і механізм вакцин, в роботі [41] – неконтрольована ШІС з обмеженими властивостями, а в роботі [45] – ШІС на основі алгоритму клонального відбору.

Для порівняння різних підходів виявлення і класифікації мережевих атак, результати робіт на основі використання набору даних DARPA/KDD-99 систематизовано у вигляді таблиці 1.1 [48–50] і визначено достовірність виявлення і класифікації мережевих атак найбільш ефективними методами (рисунк 1.3).

Таблиця 1.1 – Порівняльний аналіз різних підходів виявлення і класифікації мережевих атак на основі набору даних DARPA/KDD-99

Автор рік публікації	Технологія	Достовірність			
		DOS	Probe	R2L	U2R
[138] Sabhnani, M., 2003	Класифікатор Гауса	82,4	90,2	9,6	22,8
	K-NN	97,3	87,6	6,4	29,8
	Алгоритм найближчого кластера	97,1	88,8	3,4	2,2
	Лідер-алгоритм	97,2	83,8	1,0	6,6
	Алгоритм гіперсфери	97,2	84,8	1,0	8,3
	Fuzzy Art Map	97,0	77,2	3,7	6,1
	Дерево рішень C4.5	97,0	80,8	4,6	1,8
	Переможець KDD-99	97,1	83,3	8,4	13,2
	Agarwal and Joshi	96,9	73,0	10,7	6,6
	Multi-Classifier	97,3	88,7	9,6	29,8
[31] Cannady J., 1998	MLFF-мережа	91			
[43] Ramadas M., 2003	SOM-мережа	95.42			
[39] Moradi M., 2004	MLP-мережа	91			
[98] Siddiqui M., 2004	BP-мережа Fuzzy ARTMAP	81.37 80.52			
[40] Mukkamala S., 2005	BP-мережа	97.04			
[36] Grediaga A, 2006	MLP-мережа SOM-мережа	94.29 99.01			
[139] Kayacik H.G. та ін., 2007	Ієрархія SOM -мереж	96.9	81.3	0.0	1.1
[44] Sammany M., 2007	MLP-мережа	93.43			
[140] Liu G. та ін., 2007	PCA-мережі і SOM RBF-мережі Ієрархія PCA	99.9 98.8 100.0	75.2 98.8 100.0	77.0 97.2 97.2	– – –
[141] Wang G. та ін., 2010	Дерева рішень Байєсові мережі MLP і нечітка кластеризація	99.8 99.7 99.9	50.0 52.6 48.1	33.3 46.2 93.2	50.0 25.0 83.3
[42] Novosad T. та ін., 2010	Flexible Neural Tree	98.8	99.3	98.8	99.9
[142] Muna M.J. та ін., 2010	Fuzzy NN	100.0	100.0	99.8	40.0
[37] Ibrahim L.M., 2010	DTDNN-мережа	97.6	98.2	95.8	96.2
		Normal – 98.4			
[38] Jing Xiao-Pei та ін. 2010	ШІС (генетичний алгоритм)	99.55	98.51	97.12	97.89
[41] Murad Abdo Rassam та ін., 2010	Неконтрольована ШІС	99.69	96.40	99.31	80.00
		Normal – 97.52			
[11, 12] Golovko V. та ін., 2011	Нейронна мережа (RNN and MLP)	99.61	95.15	99.37	98.08
[45] Sridevi R. та ін., 2011	ШІС (алгоритм клонального відбору)	87.6 Normal – 38.3			

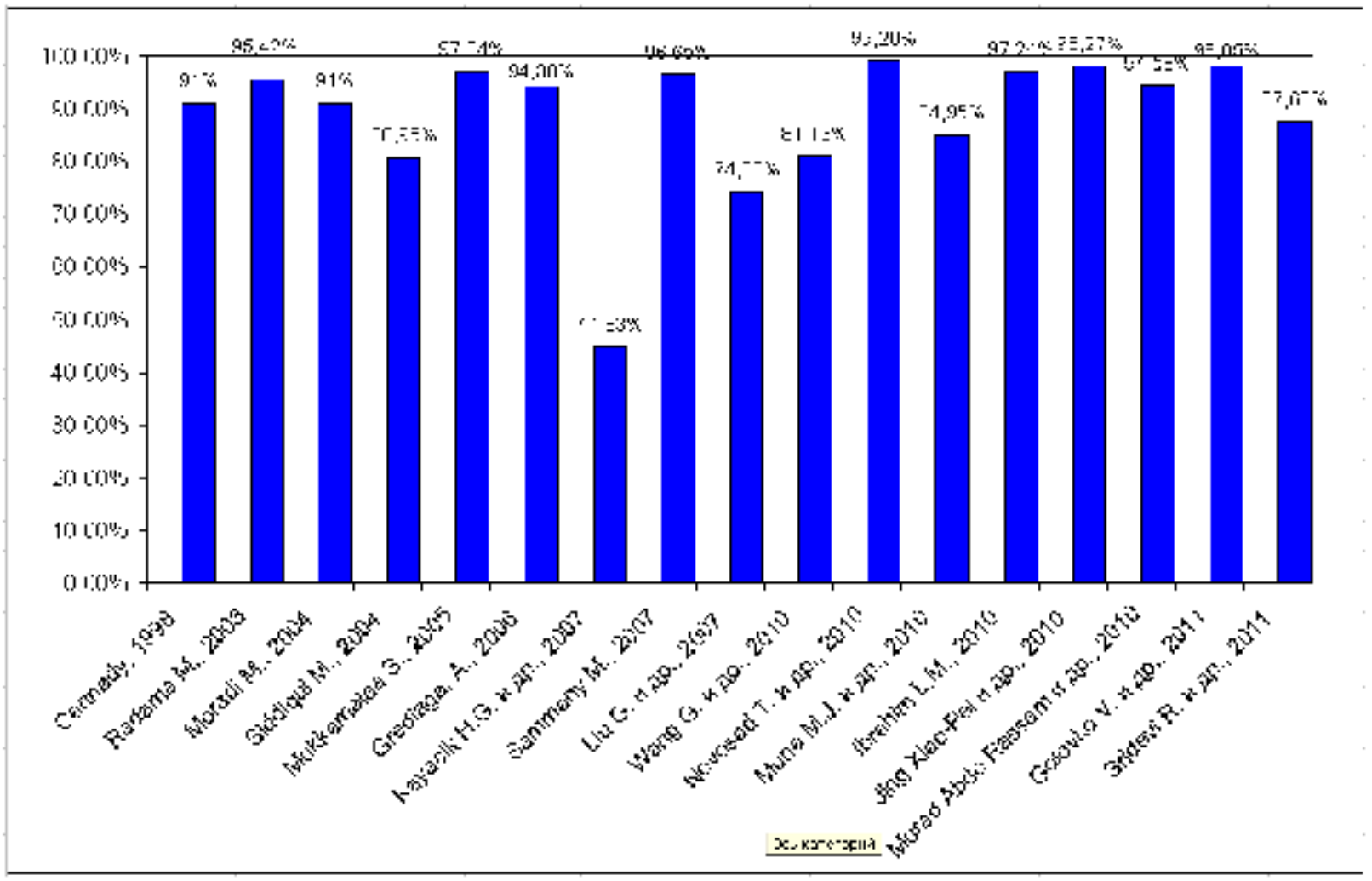


Рисунок 1.3 – Достовірність виявлення і класифікації мережевих атак найбільш ефективними методами на основі набору даних DARPA/KDD-99

Узагальнюючи проведений огляд методів штучного інтелекту для виявлення і класифікації мережевих атак, можна зробити наступні висновки:

– всі розглянуті методи мають низький відсоток виявлення і класифікації мережевих атак класу U2R і R2L;

– часто дослідники перевіряють свої моделі на обмежених наборах даних, отриманих синтетичним шляхом, що не завжди дозволяє адекватно оцінити функціональність даної моделі на реальних даних;

– велика частина відомих методів дозволяє виявляти лише відомі мережеві атаки, актуальним є виявлення раніше невідомих атак.

Як видно з приведеного вище огляду, нейромережеві методи виявлення і класифікації атак продовжують стрімко розвиватися, що підтверджує велика кількість робіт за останніх два роки в цьому напрямі [37–45, 141–142], причому основна тенденція – застосування комбінацій нейронних мереж з іншими методами.

Важливою перевагою нейронних мереж при виявленні і класифікації

мережевих атак є їх здатність «вивчати» характеристики зумисних атак та ідентифікувати елементи, які не схожі на ті, що спостерігалися в мережі раніше. Крім того, ШНМ здатні успішно аналізувати неповні або спотворені дані, що мають функції прогнозування, які дозволяють IDS робити рішучі дії до того, як атака стане успішною, а також виявляти раніше невідомі мережеві атаки, завдяки здатності до узагальнення результатів.

1.3 Аналіз наборів тестових даних і оцінка достовірності інформаційних технологій

У науково-дослідних роботах основною проблемою при спробі порівняння різних підходів до виявлення і класифікації атак є те, що практично немає двох різних досліджень, які використовують в якості даних для аналізу однакові вхідні дані. А якщо навіть і використовуються однакові набори даних, то навчання і тестування пропонованої системи все одно проводиться, найчастіше, на різних підмножинах із заявленого набору.

Оскільки, в дослідницьких роботах запропоновані підходи виявлення і класифікації мережевих атак, побудовані на основі наборів даних, то якість підготовки даних безпосередньо впливає на якість підготовки моделей. Як правило, набори тестових даних зібрані з трьох джерел: дані мережевих пакетів, дані поведінки користувачів або дані системних викликів (таблиця 1.2).

Першим поширеним набором даних став Internet Exploration Shootout Dataset (IES) [143, 144], що представляв чотири колекції даних, що включали заголовки пакетів, отримані в реальній мережі: одна з нормального трафіку і три з аномаліями.

У 1998 році була сформована база даних DARPA98, де даними є записи TCP-з’єднань реальної ЛОМ, які включають відомі і невідомі атаки. Ця база даних використовувалася дослідниками в [36, 39, 40, 43, 44].

Таблиця 1.2 – Набори тестових даних

Джерело даних	Назва набору даних	Абревіатура
Параметри мережевих з'єднань	Internet Exploration Shootout Dataset	IES
	DARPA 1998 TCPDump Files	DARPA98
	DARPA 1999 TCPDump Files	DARPA99
	KDD99 Dataset	KDD99
	10% KDD99 Dataset	KDD99-10
Поведінка користувачів	UNIX User Dataset	UNIXDS
Системні виклики	DARPA 1998 BSM Files	BSM98
	DARPA 1999 BSM Files	BSM99
	University of New Mexico Dataset	UNM

База даних KDD Cup1999 Data [81] була сформована в рамках проведення міжнародної наукової конференції KDD-99, метою якої було стимулювання досліджень в області обробки даних і створення нових алгоритмів виявлення і класифікації мережевих атак. Цей набір даних підготовлений SJ Stolfo разом із співавторами [145] і будується на основі даних, отриманих в DARPA98 [146–148]. DARPA98 включає близько 4 Гб стиснених TCP-даних мережевого трафіку, які формувалися впродовж 7 тижнів. В результаті отримано близько 5 мільйонів записів, кожен з яких розміром близько 100 байт.

База даних KDD-99 містить параметри, як нормальних мережевих з'єднань, так і 22 типів мережевих атак, що відносяться до чотирьох класів [81]. У таблиці 1.3 представлений розподіл по класах в 10% базі даних KDD-99 [81].

Таблиця 1.3 – Розподіл по класах в базі даних KDD99

Клас	Кількість записів	Відсоток
Normal	97278	19.6911%
DOS	391458	79.2391%
U2R	52	0.0105%
R2L	1126	0. 2279%
Probe	4107	0. 8313%

Співвідношення атак у відсотках по класах в базі даних KDD-99 представлено на рисунку 1.4.

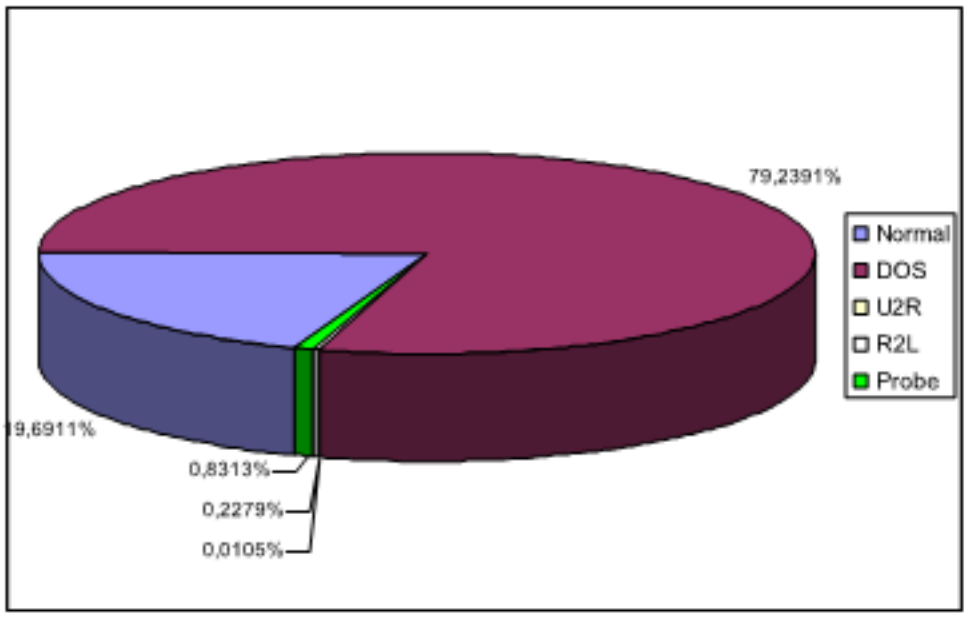


Рисунок 1.4 – Співвідношення атак по класах в базі даних KDD-99 (у відсотках)

У приведених нижче таблицях 1.4–1.7 і на рисунках 1.5–1.8 представлений розподіл атак по типах всередині класів.

Таблиця 1.4 – Розподіл по типах атак в класі DOS

Клас	Тип атак	Кількість атак
DOS	back	2203
	land	21
	neptune	107201
	pod	264
	smurf	280790
	teardrop	979

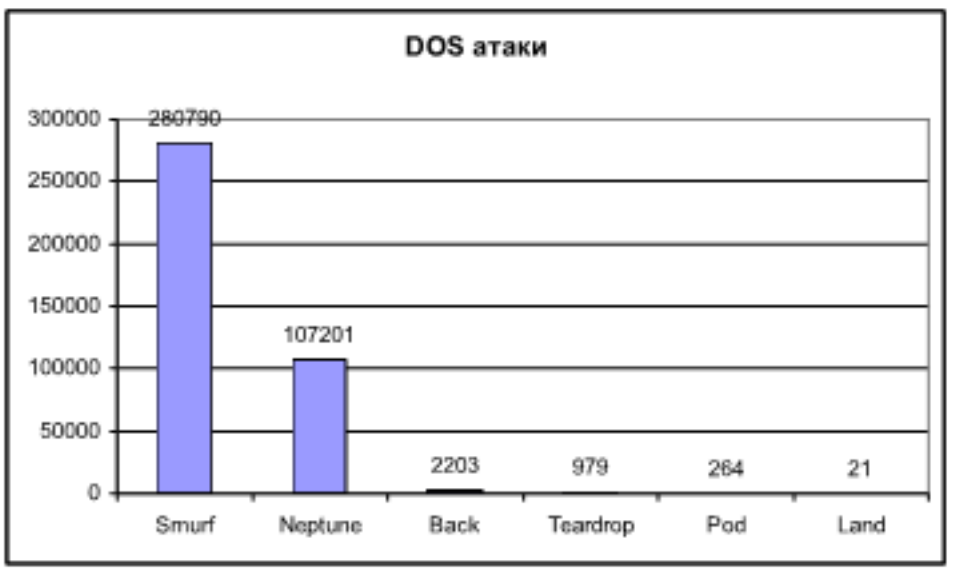


Рисунок 1.5 – Розподіл атак по типах в класі DOS

Таблиця 1.5 – Розподіл по типах атак в класі U2R

Клас	Тип атак	Кількість атак
U2R	buffer_overflow	30
	loadmodule	9
	perl	3
	rootkit	10

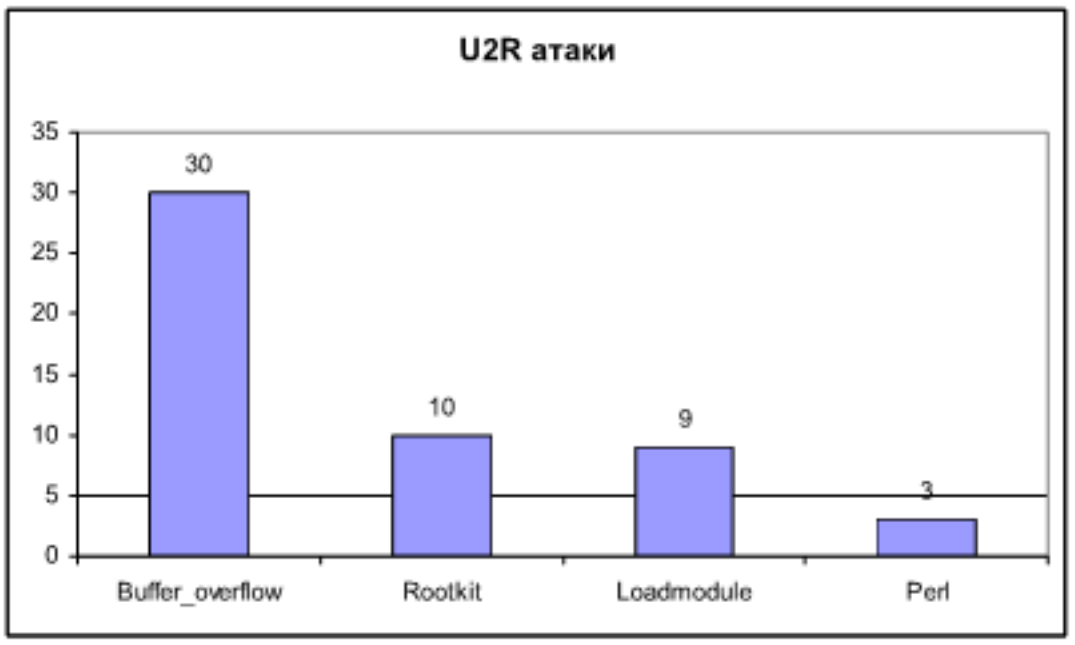


Рисунок 1.6 – Розподіл атак по типах в класі U2R

Таблиця 1.6 – Розподіл по типах атак в класі R2L

Клас	Тип атак	Кількість атак
R2L	ftp_write	8
	guess_passwd	53
	imap	12
	multihop	7
	phf	4
	spy	2
	warezclient	1020
	warezmaster	20

Таблиця 1.7 – Розподіл по типах атак в класі Probe

Клас	Тип атак	Кількість атак
Probe	Portweep	1040
	Ipsweep	1247
	Nmap	231
	Satan	1589

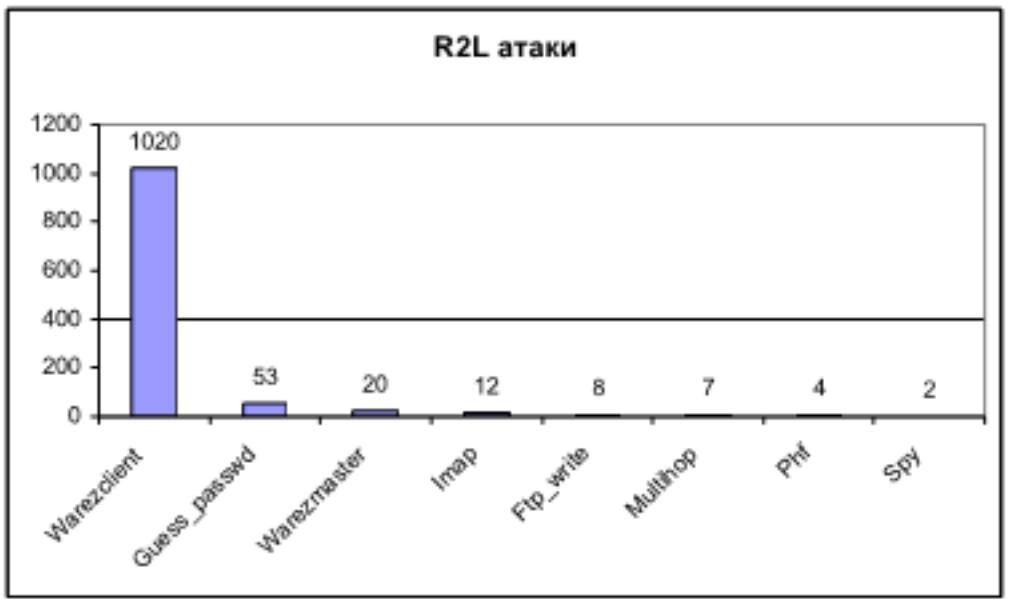


Рисунок 1.7 – Розподіл атак по типах в класі R2L

Рисунок 1.8 – Розподіл атак по типах в класі Probe

У базі даних KDD-99 виділяють 41 параметр (або атрибут) мережевого з'єднання, які, у свою чергу, об'єднані в три групи [81] (додаток Б):

1. Вбудовані параметри. Ці параметри отримані із зони заголовка мережевих пакетів. Виділяють 9 вбудованих параметрів, які містять інформацію про час роботи з'єднання, тип протоколу, кількість переданих байт і т. д.

2. Параметри контенту, отримані із зони контенту і містять таку інформацію, як: кількість невдалих спроб реєстрації, кількість невдалих спроб реєстрації в системі, кількість помилок, кількість операцій створення файлів і т. д. Існує 13 параметрів контенту.

3. Параметри трафіку. Обчислюються виходячи з попередніх з'єднань.

У свою чергу, виділяють параметри тимчасового і машинного трафіку. 19 параметрів трафіку містять наступну інформацію: кількість з'єднань до цієї ж IP-адреси, кількість з'єднань до цього ж номера порту і т. д.

На даний час KDD-99 широко використовується як один з небагатьох загальнодоступних наборів даних для мережевих систем виявлення атак. Ця база даних використовувалася в [138, 140], так і продовжує використовуватися сьогодні в [37–45, 141–142].

Інші набори даних представлені в [149, 150]. Оскільки дані набори даних мають певні недоліки, дослідники розробляють власні, наприклад, в [151–156].

Таким чином, проведений аналіз тестових наборів показує, що база даних KDD-99, яку визнали дослідники (найчастіше використовується в роботах, присвячених розробці систем виявлення і класифікації атак), дозволяє з деякою часткою ймовірності оцінити достовірність роботи різних підходів в даному напрямку.

Одним з основних критеріїв оцінки ефективності інформаційної технології виявлення і класифікації атак є достовірність результатів її роботи.

Зручним засобом оцінки достовірності є метод, заснований на аналізі так званої операційної характеристичної кривої (ROC – Receiver Operating Characteristic curve) [157], широко відомий в наукових колах, особливо за кордоном [158–160]. Традиційний ROC-аналіз передбачає порівняння операційних характеристик тесту – чутливості і специфічності [161–169]. Зазвичай в якості інтегральної характеристики для оцінки ефективності тесту використовується площа під ROC-кривою [170–172].

У роботах [160, 173] запропонований розвиток ROC-аналізу для випадку, коли вибір рішення здійснюється із понад двох варіантів (діагнозів). Удосконалення ROC-аналізу для вибору тесту з погляду очікуваних втрат від ухвалення помилкового рішення представлене в [174, 175], а в [176] показано практичне застосування цього методу. Відомий також метод оцінки ефективності тесту, заснований на аналізі кривих втрат (cost curves), який в

роботах [177, 178] розглядається як альтернатива ROC-аналізу.

Спочатку ROC-аналіз використовувався для відображення співвідношення між часткою (відсотком) виявлення мети за допомогою радарів і часткою (відсотком) помилкової тривоги [161, 162]. Пізніше ROC–аналіз знайшов застосування в діагностичних системах [163] і в машинному навчанні [164].

Для оцінки достовірності технології виявлення і класифікації атак на ІТМ скористаємося загальноприйнятими критеріями:

1. Чутливість, *Se* (*Sensitivity*).
2. Специфічність, *Sp* (*Specificity*).
3. Частка позитивних образів, класифікованих як негативні, FNR.
4. Частка помилково позитивних образів, FPR.
5. Коректність, *Accu* (*accuracy*).

ROC-крива часто використовується для оцінки результатів бінарної класифікації в машинному навчанні. При аналізі використовуються два класи даних, один з яких називається класом з позитивним результатом, а другий – класом з негативним результатом. ROC-крива показує залежність кількості вірно класифікованих позитивних образів від кількості невірно класифікованих негативних образів. Використовуючи термінологію ROC-аналізу, позитивні образи називаються істинно позитивною множиною, а негативні – помилково негативною. При аналізі використовується зміна параметра детектора, що у результаті дає те або інше розбиття образів на два класи. Такий параметр часто називають порогом, або точкою відсікання (*cut-off value*). Залежно від величини параметра виходять різні величини помилок I і II роду [179, 180].

Таблиця зв’язаності (*confusion matrix*) будується на основі результатів класифікації детектора і фактичної (об’єктивної) приналежності образів до того або іншого класу (таблиця 1.8).

Таблиця 1.8 – Таблиця зв’язаності

Клас	Приналежність образів	
	Позитивні	Негативні
Позитивні	TP	FP
Негативні	FN	TN

Від конкретного завдання залежить те, що приймати за позитивну подію, а що – за негативну. При виявленні і класифікації мережевих атак, позитивним результатом буде клас «мережева атака», а негативним – клас «нормальне з’єднання». І навпаки, якщо визначати ймовірність того, що мережеве з’єднання легітимне, то позитивним результатом буде клас «нормальне з’єднання», а негативним – «мережева атака».

Для оцінки достовірності роботи технології виявлення і класифікації атак на ІТМ, скористаємося наступними характеристиками:

– TP (*True Positives*) – вірно класифіковані позитивні образи (так звані істинно позитивні випадки), тобто кількість правильно класифікованих атак;

– TN (*True Negatives*) – вірно класифіковані негативні образи (істинно негативні випадки), кількість правильно класифікованих нормальних з’єднань;

– FN (*False Negatives*) – позитивні образи, класифіковані як негативні (помилка I роду). Це так званий «помилковий пропуск» – коли подія, що цікавить нас, помилково не виявляється (помилково негативні випадки). Тобто, кількість атак, класифікованих як нормальні з’єднання, невиявлення атаки;

– FP (*False Positives*) – кількість негативних образів, класифіковані як позитивні (помилка II роду). Це помилкове виявлення, оскільки за відсутності події помилково виноситься рішення про її наявність (помилково позитивні випадки). Тобто, кількість нормальних з’єднань, класифікованих як атаки.

При ROC-аналізі часто оперують не абсолютними показниками, а відносними, вираженими у відсотках, так званими частками. Тоді параметри,

що характеризують якість класифікації, визначаються таким чином:

– TPR (True Positives Rate) – чутливість (Sensitivity, Se), ймовірність правильної класифікації атак:

$$TPR = \frac{TP}{TP + FN} \tag{1.1}$$

– FNR (*False Negative Rate*) – помилка першого роду або рівень значущості, що характеризує ймовірність неправильної класифікації атак:

$$FNR = 1 - TPR = \frac{FN}{TP + FN} \tag{1.2}$$

– TNR (*True Negative Rate*) – специфічність (потужність критерію) (Specificity, Sp), що характеризує ймовірність правильної класифікації нормальних з'єднань:

$$TNR = \frac{TN}{FP + TN} \tag{1.3}$$

– FPR (False Positives Rate) – помилка другого роду, що характеризує ймовірність класифікації нормальних з'єднань, як атак (ймовірність помилкових спрацьовувань):

$$FPR = 1 - TNR = \frac{FP}{TN + FP} \tag{1.4}$$

Accuracy – це точність класифікації, що відображає якість роботи системи в цілому і обчислюється згідно формули:

$$Acc u = \frac{TP + TN}{TP + FN + FP + TN}. \quad (1.5)$$

Classification Rate – рівень якості класифікації для кожного класу окремо

$$CR_i = \frac{TP_i}{TP_i + FN_i}. \quad (1.6)$$

або для системи в цілому

$$CR = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FN_i)}. \quad (1.7)$$

ROC-аналіз, зокрема, зводиться до побудови ROC-кривих, що відображають зміну чутливості і специфічності залежно від зміни величини порогу детектора. Так, ROC-крива будується таким чином:

1. Для кожного значення порогу, яке змінюється від 0 до 1 з деяким заздалегідь заданим кроком, розраховуються значення чутливості *Se* і специфічності *Sp*.

2. Будується графік залежності. По осі Y відкладаються значення чутливості *Se*, а по осі X відкладаються значення *100% - Sp*.

Для ідеального класифікатора графік ROC-кривої проходить через верхній лівий кут, де частка істинно позитивних випадків складає 100% або 1.0 (ідеальна чутливість), а частка помилково позитивних прикладів рівна нулю.

ROC-аналіз застосовувався в роботах багатьох дослідників [37–45, 181–184] для оцінки достовірності запропонованих рішень.

1.4 Напрями дослідження і постановка задачі

Порівняльний аналіз існуючих інформаційних технологій виявлення і класифікації атак на ІТМ дозволив виявити недостатню достовірність результатів їх роботи. На основі проведеного аналізу найбільш поширених інформаційних технологій виявлення і класифікації мережевих атак, можна зробити наступні висновки:

– сигнатурний і статистичний аналіз показав нездатність виявляти нові або невідомі атаки, які характеризуються відсутністю записів в системі про них;

– евристичні методи мають високу ймовірність помилкових спрацювань.

Така ситуація стимулює пошук і розробку нових методів та інформаційних технологій, спрямованих на підвищення достовірності виявлення і класифікації атак на ІТМ.

Одним з перспективних напрямків є застосування методів штучного інтелекту. Разом з тим, дані підходи характеризуються наявністю ряду вузьких місць, таких як складність створення або вибору необхідних детекторів атак, складність адаптації до невідомих атак, здатність коректно працювати тільки на невеликих наборах даних.

Для розв’язання цих проблем запропоновано використати інтеграцію методів штучних нейронних мереж і штучних імунних систем, оскільки вони поєднують високу достовірність виявлення і класифікації атак з властивістю до навчання, адаптації, самоорганізації і пам’яті. Це сучасний апарат підтримки прийняття рішень, який дає можливість враховувати невизначеності статистичного і структурного характеру при моделюванні складних систем. Крім того, обидва наукові напрями дуже швидко розвиваються і знаходять все більше практичних застосувань. Це свідчить про достатньо високу ефективність штучних нейронних мереж і штучних імунних систем при вирішенні практичних задач.

На основі проведеного аналізу для вирішення задачі підвищення достовірності виявлення і класифікації атак на ІТМ запропоновано створити нову інформаційну технологію на базі теорії штучних нейронних мереж і штучних імунних систем.

В якості тестового набору запропоновано використати базу даних KDD, яка найчастіше використовується в роботах, присвячених розробці сучасних технологій виявлення і класифікації мережевих атак, що дозволило з деякою часткою ймовірності оцінити достовірність роботи різних підходів у даному напрямку.

Для проведення статистичної оцінки достовірності розробленої технології запропоновано використати ROC-аналіз.

Все вище сказане дозволяє поставити в дисертаційній роботі наступні наукові задачі:

1. Аналіз існуючих інформаційних технологій для визначення шляхів підвищення достовірності виявлення та класифікації атак на інформаційні телекомунікаційні мережі.

2. Розробка узагальненої функціональної моделі обробки інформації при виявленні і класифікації атак на інформаційні телекомунікаційні мережі.

3. Вибір нейронної мережі для виявлення і класифікації атак на інформаційні телекомунікаційні мережі.

4. Розробка методу побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі на базі нейронної мережі, що характеризується малим обсягом навчальної вибірки.

5. Розробка методу побудови сукупного класифікатора для ієрархічної класифікації атак на інформаційні телекомунікаційні мережі на основі багатоканальних нейромережевих детекторів.

6. Розробка комбінованого методу виявлення і класифікації атак на інформаційні телекомунікаційні мережі на основі інтеграції нейромережевих детекторів в штучну імунну систему.

7. Створення інтелектуальної інформаційної технології для вирішення

задачі виявлення та класифікації атак на інформаційні телекомунікаційні мережі на основі запропонованих методів.

8. Проведення статистичної оцінки достовірності розробленої інформаційної технології з метою порівняння з відомими рішеннями.

РОЗДІЛ 2
НЕЙРОМЕРЕЖЕВІ МЕТОДИ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ

2.1 Узагальнена функціональна модель прийняття рішень при виявленні і класифікації атак

Як вже наголошувалося, виявлення і класифікація мережевих атак на комп'ютерну систему відбувається на основі аналізу інформації, що передається по каналах передачі даних ІТМ. Узагальнену функціональну модель прийняття рішень при виявленні і класифікації атак на ІТМ можна представити у вигляді наступних етапів (рисунок 2.1):

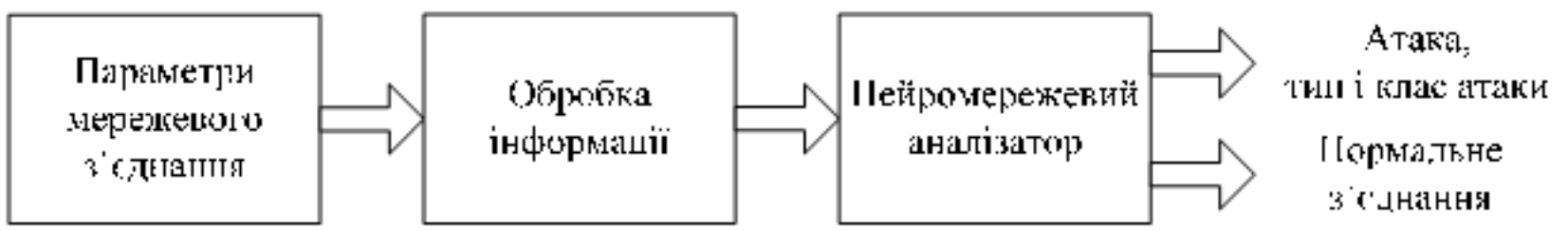


Рисунок 2.1 – Узагальнена функціональна модель прийняття рішень при виявленні і класифікації атак на ІТМ

На першому етапі, за допомогою спеціалізованого програмного забезпечення [189] відбувається захоплення параметрів мережевого з'єднання, і з кожного мережевого пакету виділяється 41 параметр. Оскільки типи параметрів, тобто значення, які вони можуть приймати, різні (наприклад, параметр *duration* відповідає за час роботи в секундах встановленого з'єднання і приймає цілочисельне значення, а параметр *protocol type* містить тип протоколу, по якому йде обмін інформацією і приймає символьне значення), то для аналізу інформації за допомогою нейронних мереж, значення параметрів необхідно трансформувати в числові значення.

Так, наприклад, якщо початкове значення *protocol type* дорівнює *tcp*, то після перетворення воно дорівнюватиме 1. Перетворення символьних

значень параметра *service* представлене в таблиці 2.1.

Таблиця 2.1 – Перетворення символьних значень параметра *service* мережевого з'єднання

Значення до перетворення	Значення після перетворення	Значення до перетворення	Значення після перетворення	Значення до перетворення	Значення після перетворення
http	1	pm_dump	21	supdup	41
smtp	2	mtp	22	iso_tsap	42
finger	3	remote_job	23	csnet	43
domain_u	4	s10	24	ldap	44
telnet	5	name	25	10el	45
eco_i	6	whois	26	k47	46
ntp_u	7	nnsp	27	klogin	47
auth	8	exec	28	echo	48
other	9	printer	29	daytime	49
ftp	10	efs	30	netstat	50
login	11	courier	31	uucp_path	51
private	12	discard	32	irc	52
imap4	13	systat	33	urp_i	53
time	14	pop	34	x11	54
rje	15	sunrpc	35	tim_i	55
link	16	nntp	36	urh_i	56
ctf	17	netbios	37	tftp_u	57
uucp	18	sqlnet	38	red_i	58
hostnames	19	vmnet	39	ecr_i	59
gopher	20	bgp	40		

Адаптація параметрів з'єднання до подачі на вхід нейронної мережі представлена в додатку В.

Приклади створення векторів для навчання нейронної мережі приведені в додатку Г.

Перед подачею даних на вхід нейромережевого аналізатора необхідно провести їх нормалізацію. Нехай діапазон зміни вхідної змінної рівний [x_{min}, x_{max}]. Тоді для перетворення вхідних даних в прийнятний діапазон значень [a,b]

$$\ddot{x}_i^k = \frac{(x_i^k - x_{\min})(b-a)}{(x_{\max} - x_{\min})} + a\,, \qquad (2.1)$$

де x_i^k – i-а компонента k-го вектора.

На останньому етапі нейромережевий аналізатор здійснює виявлення і класифікацію атак. Завдяки своїм особливостям, таким як високий ступінь паралелізму обробки інформації, здатність до узагальнення, адаптація до змін навколишнього середовища, розпізнавання зашумленних образів і т. д., нейронні мережі дозволяють досягти хороших результатів у вирішенні таких складних інженерних завдань, як розпізнавання образів, класифікація, прогнозування, системи контролю і т. д.

У зв'язку із здатністю ШНМ в процесі навчання виявляти складні залежності між вхідними і вихідними даними, які були відсутні в навчальній вибірці, і здатністю коректно класифікувати зашумлені образи, вони є привабливим інструментом для вирішення складних різноманітних задач захисту комп'ютерної інформації [31, 47, 58, 61].

Все це послужило основою для вибору ШНМ в якості основного елементу для побудови нейромережевого аналізатора. При цьому нейромережевий аналізатор може складатися з множини нейронних мереж, об'єднаних в єдину систему. Назвемо окрему нейронну мережу для виявлення і класифікації мережевих атак нейромережевим детектором. Тоді система виявлення і класифікації атак є сукупністю нейромережевих детекторів.

Представимо нейромережевий детектор (НД) у вигляді чорного ящика, який має n-входів і два виходи (рисунок 2.2).

Рисунок 2.2 – Нейромережевий детектор

Простір вихідних значень детектора можна представити в табличному вигляді (таблиця 2.2).

Таблиця 2.2 – Простір вихідних значень детектора

Y1	*Y2*	Клас
1	0	Атака
0	1	Нормальне з’єднання
0	0	Не визначено
1	1	Не визначено

Тоді першочерговою задачею проектування нейромережевого аналізатора є вибір структури нейромережевого детектора для виявлення і класифікації атак, здатного швидко навчатися і потребуючого малу кількість системних ресурсів.

2.2 Вибір базової архітектури нейромережевого детектора

Існує велика кількість різноманітних архітектур нейронних мереж, які використовуються для вирішення тих або інших складних інженерних задач. Так багатошарові персептрони [186] характеризуються прямим розповсюдженням вхідного сигналу від шару до шару і складаються з множини вхідних нейронних елементів, одного або декількох прихованих шарів нейронних елементів і вихідного шару. Однією з головних переваг таких мереж є можливість вирішувати задачі, що важко формалізуються, або задачі, для яких алгоритмічне рішення невідоме, але для яких можливо скласти репрезентативний набір прикладів з відомими рішеннями. MLP при навчанні, за рахунок своєї внутрішньої будови, виявляє закономірності і взаємозв’язки між вхідними і вихідними образами, узагальнюючи отриманий на навчальній вибірці досвід.

Нейронні мережі з радіально-базисною функцією активації (radial basis function networks - RBF) [186] застосовуються для вирішення задач прогнозування, апроксимації функцій, розпізнавання образів і т. д.

Нейронні мережі Кохонена дозволяють в результаті навчання здійснювати типологічно безперервне відображення *F*-вхідного *n*-мірного простору у вихідний *m*-мірний простір. Структура такої нейронної мережі є мережею з прямим розповсюдженням сигналу. В якості методу навчання використовується конкурентне навчання. У міру надходження вхідних образів на таку мережу за допомогою навчання відбувається розбиття *n*-мірного вхідного простору на різні області рішень, кожній з яких відповідає окремий нейрон.

LVQ мережа є розширенням мережі Кохонена і містить, окрім конкурентного шару, лінійний шар, який здійснює класифікацію кластерів, виділених шаром Кохонена.

Обгрунтуємо вибір класу нейронної мережі для нейромережевого детектора виявлення і класифікації мережевих атак [62]. До системи виявлення вторгнень ставиться ряд жорстких вимог, однією з яких є функціонування в режимі реального часу. В результаті, необхідно мінімізувати часові витрати, пов'язані з навчанням нейронної мережі. Також, кількість атак що містяться в базі даних KDD-99, для деяких типів атак є дуже малою. Наприклад, для класу *U2R* існує всього 52 записи атак. Тому необхідно вибрати таку архітектуру мережі, яка характеризувалася б малим розміром навчальної вибірки і, відповідно, мінімальним часом навчання.

Розглянемо характеристики MLP [186, 187]. Багатошарові персептрони навчаються за допомогою алгоритму зворотного розповсюдження помилки (back-propagation algorithm) [186, 187] і успішно застосовуються для вирішення багатьох складних завдань класифікації, розпізнавання та ін. У [187] зазначено, що на здатність нейронної мережі до коректного узагальнення впливають розмір навчальної вибірки і архітектура нейронної мережі.

Для коректного навчання нейронній мережі досить, щоб розмір навчальної вибірки *L* задовільняв наступне співвідношення [187]:

$$L \approx O(W/\varepsilon), \tag{2.2}$$

де W – загальна кількість параметрів, що налаштовуються (вагових коефіцієнтів і порогових значень);

ε - допустима помилка класифікації;

$O()$ – порядок величини, тобто, наприклад, для помилки в 5% кількість прикладів навчання повинна в 5 разів перевершувати кількість вільних параметрів мережі W.

Загальна кількість параметрів, що налаштовуються, обчислюється згідно наступного виразу:

$$W = m(n+k+1)+k, \tag{2.3}$$

де n – кількість вхідних нейронів;

m – кількість нейронів прихованого шару;

k – кількість нейронів вихідного шару.

Як приклад, проведемо розрахунок розміру навчальної вибірки для MLP, RBF і LVQ для наступної структури нейронної мережі:

– кількість вхідних нейронів повинна дорівнювати кількості параметрів мережевого з'єднання, тобто $n = 41$;

– кількість прихованих нейронів $m = 10$;

– кількість вихідних нейронів $k = 2$, тобто кожен з вихідних нейронів відображає той або інший клас вхідного образу.

В результаті, у разі застосування MLP в якості детектора для виявлення і класифікації мережевих атак навчальна вибірка для навчання нейронної мережі з допустимою помилкою класифікації $\varepsilon=0,1$ повинна складатися, згідно з виразами (2.2) і (2.3) з *4420* образів, що є неприйнятним для багатьох класів атак.

Проведемо аналогічний аналіз для RBF мереж [186, 187]. Мережі на основі радіальних базисних функцій також є багатошаровими нейронними

мережами. Перший шар таких мереж є вхідним шаром і забезпечує зв'язок мережі із зовнішнім середовищем. Другий шар – прихований шар, виконує нелінійне перетворення вхідного простору образів в прихований простір відповідно до радіально-базисної функції активації, який часто має істотно вищу розмірність, ніж вхідний простір. Третій шар – вихідний, він складається з лінійних нейронів. Згідно [187], для розмірності навчальної вибірки L оптимальна кількість прихованих нейронних елементів для забезпечення мінімальної помилки узагальнення повинна бути наступною: $L^{1/3}=<m<=L$. Тому RBF мережа підходить для навчання різних типів атак, проте кількість нейронів прихованого шару в такій мережі зростає пропорційно розміру навчальної вибірки.

Тепер розглянемо нейронну мережу векторного квантування (LVQ) [186, 187] з ідентичною кількістю нейронів в кожному з шарів. У прихованому шарі такої мережі використовуємо нейронні елементи Кохонена [186–188]. Для навчання такої мережі досить, щоб на кожен нейрон прихованого шару (кластер) припадало хоч би два образи з навчальної вибірки. Відповідно до цього, розмір навчальної вибірки для LVQ мережі можна визначити згідно наступного виразу:

$$L>= 2m. \tag{2.4}$$

Як видно, для навчання мережі LVQ з десятьма нейронами Кохонена в прихованому шарі необхідно мати навчальну вибірку розмірністю більшою *20* образів. Таким чином, LVQ мережа є хорошим кандидатом в якості нейромережевого детектора з погляду мінімізації розмірності навчальної вибірки і, відповідно, часу навчання [62].

Щоб зробити остаточний вибір, проведемо практичні експерименти з метою оцінки якості виявлення і класифікації мережевих атак нейромережевими детекторами на базі трьох архітектур – MLP, RBF і LVQ. Експерименти проводилися в Matlab.

Програмна реалізація детекторів на базі нейронних мереж MLP, RBF і LVQ представлена в додатках Д, Е і Ж відповідно.

Для експериментів використовуються атаки типів *dos_back* і *dos_neptune*.

У таблиці 2.3 представлені результати виявлення вибраних типів атак нейромережевим детектором на базі MLP, в таблиці 2.4 представлені результати виявлення мережевих атак нейромережевим детектором на базі LVQ з нейронами Кохонена в прихованому шарі, в таблиці 2.5 представлені результати виявлення мережевих атак нейромережевим детектором на базі RBF і в таблиці 2.6 представлені узагальнені результати.

Таблиця 2.3 – Результати виявлення мережевих атак на базі MLP

DoS_back (розмір тестової вибірки – 2139 атак)				
	TPR (Se),%	TNR (Sp),%	FNR,%	FPR,%
Детектор MLP1	99,05	98,27	0,95	0,73
Детектор MLP2	99,14	96,87	0,86	3,13
Детектор MLP $_3$	99,05	97,62	0,95	2,38
Детектор MLP $_4$	**99,05**	**99,41**	**0,95**	**0,59**
Детектор MLP $_5$	100.0	87,32	0,0	12,68
Детектор MLP $_6$	99,05	99,37	0,95	0,63
Детектор MLP $_7$	99,18	90,54	0,82	9,46
Детектор MLP $_8$	99,59	97,56	0,41	2,44
Детектор MLP $_9$	99,05	98,79	0,95	1,21
Детектор MLP $_{10}$	99,05	99,02	0,95	0,98
DoS_neptune (розмір тестової вибірки – 107137 атак)				
Детектор MLP $_{11}$	99,99	95,64	0,01	4,36
Детектор MLP $_{12}$	**99,98**	**99,54**	**0,02**	**0,46**
Детектор MLP $_{13}$	99,81	95,17	0,19	4,83
Детектор MLP $_{14}$	99,99	93,26	0,01	6,74
Детектор MLP $_{15}$	99,99	94,54	0,01	5,46
Детектор MLP $_{16}$	100,0	93,18	0,0	6,82
Детектор MLP $_{17}$	99,98	98,52	0,02	1,48
Детектор MLP $_{18}$	99,99	96,00	0,01	4,00
Детектор MLP $_{19}$	100,0	98,06	0,0	1,94
Детектор MLP $_{20}$	100,0	96,72	0,0	3,28

Таблиця 2.4 – Результати виявлення мережевих атак на базі LVQ

DoS_back (розмір тестової вибірки – 2139 атак)				
	TPR (Se),%	TNR (Sp),%	FNR,%	FPR,%
Детектор Koh $_1$	99,05	97,00	0,95	3,01
Детектор Koh $_2$	100,0	88,63	0,0	11,37
Детектор Koh $_3$	**99,00**	**99,80**	**1,00**	**0,20**
Детектор Koh $_4$	99,05	98,14	0,95	1,86
Детектор Koh $_5$	99,37	90,46	0,63	9,54
Детектор Koh $_6$	100,0	85,73	0,0	14,27
Детектор Koh $_7$	100,0	94,0	0,0	6,0
Детектор Koh $_8$	99,05	98,44	0,95	1,56
Детектор Koh $_9$	99,05	91,41	0,95	8,59
Детектор Koh $_{10}$	99,77	90,49	0,23	9,51
DoS_neptune (розмір тестової вибірки – 107137 атак)				
Детектор Koh $_{11}$	100,0	94,53	0.0	5,47
Детектор Koh $_{12}$	**100,0**	**99,90**	**0,0**	**0,10**
Детектор Koh $_{13}$	100,0	95,56	0,0	4,44
Детектор Koh $_{14}$	100,0	94,60	0,0	5,40
Детектор Koh $_{15}$	100,0	95,64	0,0	4,36
Детектор Koh $_{16}$	100,0	99,15	0,0	0,85
Детектор Koh $_{17}$	100,0	94,77	0,0	5,23
Детектор Koh $_{18}$	100,0	94,79	0,0	5,21
Детектор Koh $_{19}$	**100,0**	**99,73**	**0,0**	**0,27**
Детектор Koh $_{20}$	100,0	94,06	0,0	5,94

Таблиця 2.5 – Результати виявлення мережевих атак на базі RBF

DoS_back (розмір тестової вибірки – 2139 атак)				
	TPR (Se),%	TNR (Sp),%	FNR,%	FPR,%
Детектор RBF1	99,05	74,32	0,95	25,68
Детектор RBF $_2$	100,0	84,91	0,0	15,09
Детектор RBF $_3$	99,09	90,78	0,91	9,22
Детектор RBF $_4$	100,0	89,85	0,0	10,15
Детектор RBF $_5$	99,09	74,55	0,91	25,45
Детектор RBF $_6$	100,0	89,41	0,0	10,59
Детектор RBF $_7$	**99,09**	**93,72**	**0,91**	**6,28**
Детектор RBF $_8$	100,0	89,48	0,0	10,52
Детектор RBF $_9$	99,09	93,52	0,91	6,48
Детектор RBF $_{10}$	100,0	87,62	0,0	12,38
DoS_neptune (розмір тестової вибірки – 107137 атак)				
Детектор RBF $_{11}$	100,0	94,71	0,0	5,29
Детектор RBF $_{12}$	100,0	91,33	0,0	8,67
Детектор RBF $_{13}$	100,0	88,23	0,0	11,77

Продовження таблиці 2.5

Детектор RBF $_{14}$	100,0	86,05	0,0	13,95
Детектор RBF $_{15}$	100,0	85,46	0,0	14,54
Детектор RBF $_{16}$	100,0	97,29	0,0	2,71
Детектор RBF $_{17}$	100,0	90,94	0,0	9,06
Детектор RBF $_{18}$	**100,0**	**97,91**	**0,0**	**2,09**
Детектор RBF $_{19}$	100,0	97,28	0,0	2,72
Детектор RBF $_{20}$	100,0	94,19	0,0	5,81

Таблиця 2.6 – Узагальнені результати виявлення мережевих атак

DoS_back розмір тестової вибірки – 2139 атак)				
	TPR (Se),%	TNR (Sp),%	FNR,%	FPR,%
Детектор Koh $_3$	99,00	99,80	1,00	0,20
Детектор MLP $_4$	99,05	99,41	0,95	0,59
Детектор RBF $_7$	99,09	93,72	0,91	6,28
DoS_neptune (розмір тестової вибірки – 107137 атак)				
Детектор Koh $_{12}$	100,0	99,90	0,0	0,10
Детектор MLP $_{12}$	99,98	99,54	0,02	0,46
Детектор RBF $_{18}$	100,0	97,91	0,0	2,09

Як видно з представлених таблиць, найкращі результати показала нейронна мережа LVQ, де в прихованому шарі застосовуються нейрони Кохонена. Достовірність виявлення (чутливість нейромережевого детектора) атак класу *dos_back* для неї склав 99,00% при рівні помилок другого роду 0,2% (детектор Koh_3). Ця ж нейронна мережа показала результат виявлення атак класу *dos_neptune*, рівний 100% при рівні помилок другого роду 0,1% (детектор Koh_{12}).

Кращим результатом виявлення представлених атак для мережі MLP став 99,05% для атак класу *dos_back* (детектор MLP_4) при рівні помилок другого роду 0,59%, а для атак класу *dos_land* ця мережа показала результат рівний 99,98% при рівні помилок другого роду – 0,46% (детектор MLP_{12}).

Нейронна мережа RBF показала наступні результати: 99,09% для *dos_back* при рівні помилок другого роду 6,28% (детектор RBF_7), і 100% для *dos_neptune* при рівні помилок другого роду 2,09% (детектор RBF_{18}).

На рисунках 2.3 і 2.4 представлено порівняльний аналіз вибору основи нейромережевого детектора з використанням ROC-аналізу.

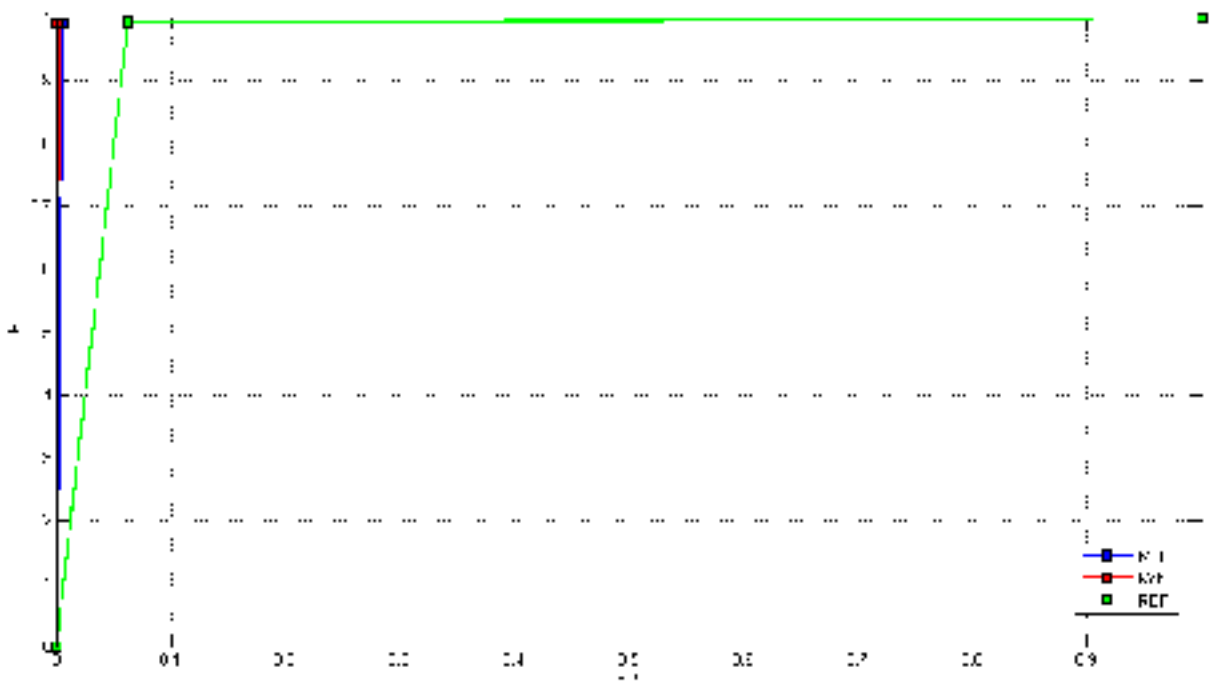


Рисунок 2.3 – Залежність FPR і TPR при виявленні атак dos_back різними нейронними мережами

На рисунку 2.3 зображена залежність помилки другого роду, коли нормальні з’єднання класифікуються як атаки (FPR), і чутливості – ймовірності правильної класифікації атак (TPR) при виявленні атак dos_back різними нейронними мережами.

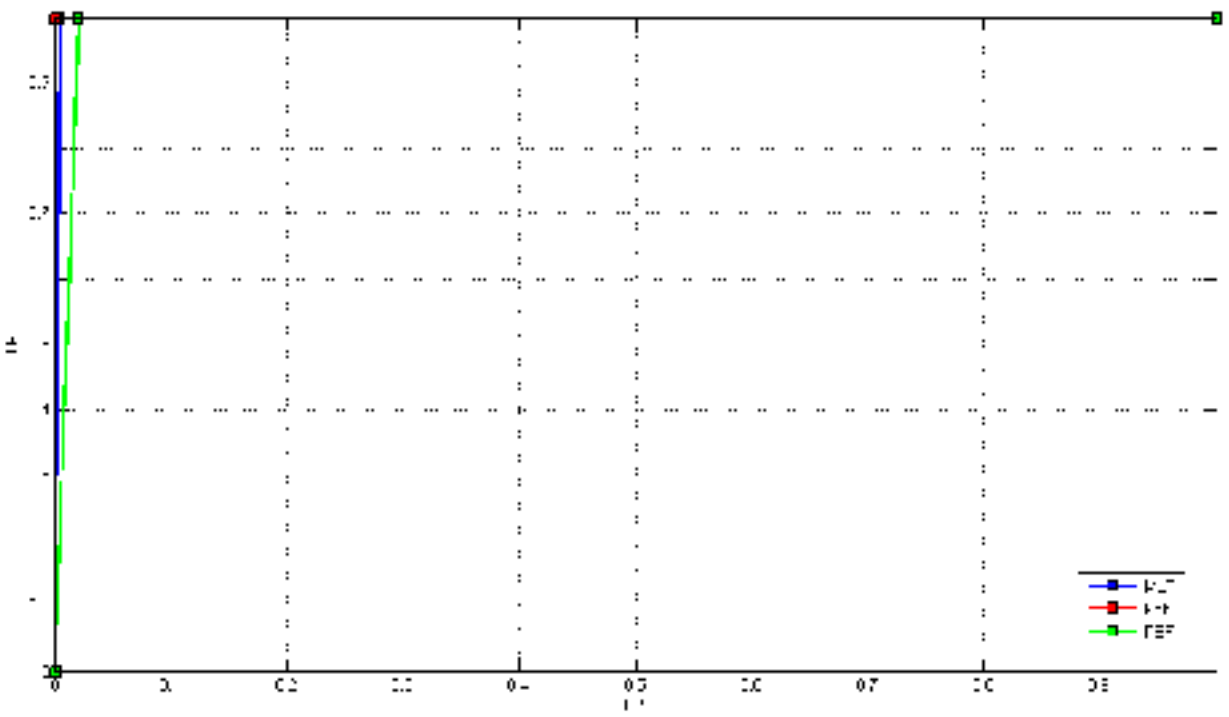


Рисунок 2.4 – Залежність FPR і TPR при виявленні атак dos_neptune різними нейронними мережами

На рисунку 2.4 зображена залежність помилки другого роду, коли нормальні з'єднання класифікуються як атаки (FPR), і чутливості – ймовірності правильної класифікації атак (TPR) при виявленні атак *dos_neptune* різними нейронними мережами.

На представлених рисунках синім кольором позначений детектор на базі MLP, червоним – детектор на базі LVQ і зеленим – детектор на базі RBF. Як видно, найкращі результати показує детектор на базі LVQ.

Виходячи з проведеного аналізу, в якості основи нейромережевого детектора вибрано нейронну мережу векторного квантування LVQ з нейронами Кохонена в прихованому шарі, яка характеризується малим об'ємом навчальної вибірки, що дозволяє навчити нейромережеві детектори на атаках, які характеризуються малою кількістю записів в базі.

2.3 Метод побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі

У попередньому підрозділі в якості основи для побудови нейромережевого детектора вибрано LVQ мережу. У даному підрозділі розглядається проектування нейромережевого детектора на базі нейронної мережі векторного квантування.

2.3.1 Структура нейромережевого детектора

Розглянемо проектування нейромережевого детектора на базі нейронної мережі LVQ з метою підвищення достовірності виявлення атак на ІТМ.

Перший шар нейронних елементів є розподільним і призначений для розподілу вхідних сигналів на нейрони прихованого шару. Вхідними сигналами є параметри мережевого з'єднання [81], які характеризують мережевий трафік і містять інформацію про час з'єднання, тип протоколу, кількість переданих байт, кількість виникнення помилок під час з'єднання і т.

д. Кількість нейронних елементів розподільного шару дорівнює кількості атрибутів мережевого з'єднання, тобто $n = 41$.

Другий шар нейронної мережі складається з нейронів Кохонена [186]. Шар Кохонена відіграє ключову роль в класифікації інформації і здійснює кластеризацію вхідного простору образів, внаслідок чого утворюються кластери різних образів, кожному з яких відповідає свій нейронний елемент. Для навчання нейронів прихованого шару використовується конкурентний принцип навчання відповідно до правила «переможець бере все» (winner-take-all) [186, 187]. Кількість нейронів в шарі Кохонена дорівнює m, вони пов'язані з двома нейронами вихідного шару. Розіб'ємо простір нейронів прихованого шару на два класи: клас атак і клас нормальних з'єднань. При цьому перші f нейронів шару Кохонена відповідають класу мережевих атак і пов'язані з першим нейроном вихідного шару, а останні l нейронів відповідають класу нормальних мережевих з'єднань і пов'язані з другим нейроном вихідного шару. Таким чином, кількість нейронів m в прихованому шарі Кохонена дорівнює

$$m = f + l, \tag{2.5}$$

де f – кількість перших нейронів шару Кохонена, які відповідають класу мережевої атаки;

l – кількість останніх нейронів шару Кохонена, активність яких характеризує клас нормального, легітимного з'єднання.

Таке розбиття дозволяє окремо здійснити кластеризацію атак і нормальних з'єднань в прихованому шарі. В результаті, можна отримати більш точне розділення одного класу від іншого.

Нейрони прихованого шару функціонують за принципом «переможець бере все». Відповідно до цього, під час поступлення вхідного образу необхідно обчислити для кожного нейрона евклідову відстань D_i між вхідним X і ваговим ω_i вектором відповідного нейронного елемента:

$$D_i = |X - \omega_i| = \sqrt{(X_1 - \omega_{1i})^2 + (X_2 - \omega_{2i})^2 + ... + (X_n - \omega_{ni})^2}, \qquad (2.6)$$

де $i=1,m$.

Потім визначається нейрон-переможець з номером k, який має максимальне значення эвклідової відстані

$$D_k = \max_i D_i. \qquad (2.7)$$

Вихідне значення нейрона-переможця дорівнює 1, а решти нейронних елементів – 0

$$Y_i = \begin{cases} 1, i = k \\ 0, \text{інакше} \end{cases} \qquad (2.8)$$

Третій шар складається з двох лінійних нейронних елементів, які використовують лінійну функцію активацію [186, 187] і здійснюють відображення кластерів, сформованих шаром Кохонена, в два класи, які характеризують нормальне з'єднання або атаку.

Структура нейромережевого детектора [55, 60] виявлення атак представлена на рисунку 2.5.

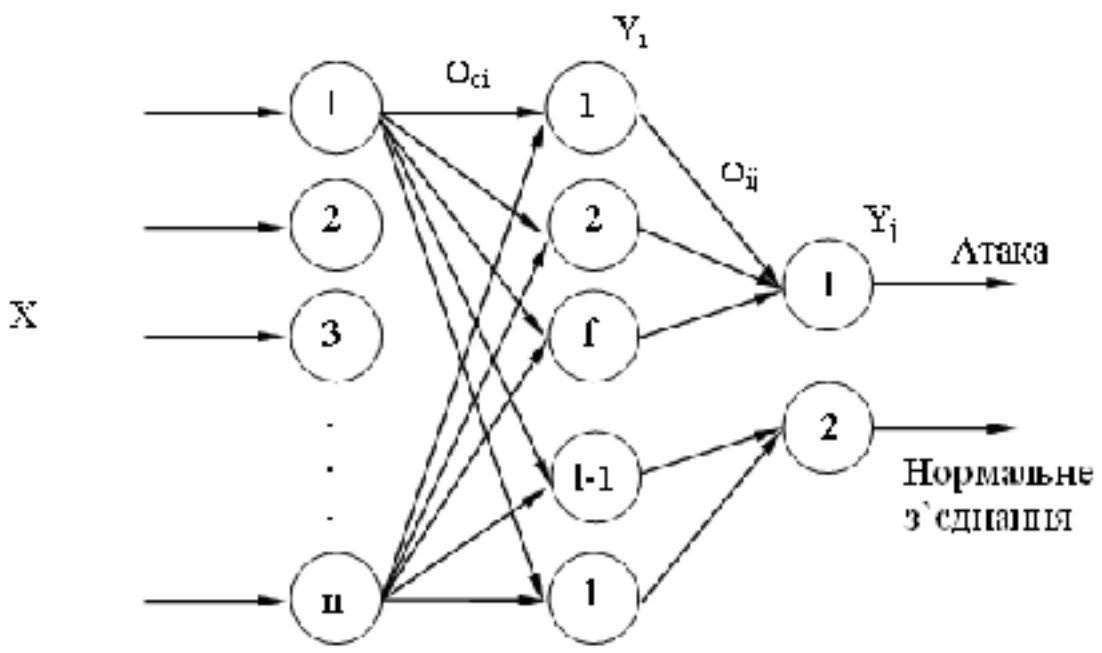


Рисунок 2.5 – Структура нейромережевого детектора для виявлення атак

Активність вихідних нейронів характеризує мережеву атаку або нормальне, легітимне з'єднання:

$$Y_1 = \begin{cases} 1, & \textit{якщо атака} \\ 0, & \textit{інакше} \end{cases}$$
$$Y_2 = \begin{cases} 1, & \textit{якщо нормальне з`єднання} \\ 0, & \textit{інакше} \end{cases} \qquad (2.9)$$

У загальному випадку вихідне значення *j*-го нейрона третього шару визначається таким чином:

$$Y_j = \sum_{i=1}^{m} \omega_{ij} \cdot Y_i, \qquad (2.10)$$

де ω_{ij} – ваговий коефіцієнт між *i*-м нейроном шару Кохонена і *j*-м нейроном лінійного шару. Всі вагові коефіцієнти мають одиничне значення.

Якщо нейрон-переможець в шарі Кохонена має номер *k*, то вихідне значення *j*-го нейрона третього шару дорівнює

$$Y_j = \omega_{kj} \cdot Y_k . \qquad (2.11)$$

Таким чином, відмінною особливістю представленого детектора є те, що нейрони в прихованому шарі розділені на дві групи: перша група характеризує клас мережевих з'єднань, що відносяться до мережевих атак; друга група нейронів характеризує клас нормальних з'єднань [55, 60]. В результаті, можна отримати більш точне розділення одного класу від іншого.

У даному параграфі представлена структура нейромережевого детектора виявлення атак, який характеризується малим обсягом навчальної вибірки і дозволяє окремо здійснити кластеризацію атак і нормальних з'єднань в прихованому шарі.

2.3.2 Навчання нейромережевого детектора

Для того, щоб нейромережевий детектор коректно функціонував – розпізнавав нормальні з'єднання і мережеві атаки – його необхідно навчити. Існують різноманітні правила навчання різних архітектур нейронних мереж [186, 187], що дозволяють досягти прийнятних результатів у вирішенні тієї або іншої задачі. Методи, згідно яких навчаються нейронні мережі, залежать від архітектури нейронної мережі і від задач, які дана мережа вирішує. Так як у нас є еталонні вихідні значення, то для навчання нейромережевого детектора будемо використовувати контрольоване конкурентне навчання [186] відповідно до правила «переможець бере все». Суть даного методу навчання полягає в тому, що в процесі навчання відбувається конкуренція між нейронними елементами шару Кохонена, внаслідок чого визначається нейронний елемент-переможець, який і характеризує приналежність до того або іншого класу даних, які розглядаються. Для виявлення нейрона-переможця використовується евклідова відстань. В процесі навчання синаптичні зв'язки для нейрона-переможця посилюються при коректній класифікації і послаблюються при некоректній класифікації вхідного образу. Для решти нейронів вагові коефіцієнти не змінюються. Таким чином, після навчання нейронної мережі, при подачі вхідного образу активність нейрона-переможця дорівнює одиниці, а решта нейронів «скидаються» в нуль.

Відмінною особливістю представленого в попередньому параграфі нейромережевого детектора є поділ нейронів в шарі Кохонена за класами, які характеризують або легітимне з'єднання, або мережеву атаку. В результаті коректна класифікація вхідного образу відбувається в тому випадку, коли при подачі на вхід нейронної мережі параметрів з'єднання, що відносяться до класу мережевої атаки, переможцем є один з f нейронів шару Кохонена, або, якщо при подачі на вхід нейронної мережі параметрів нормального з'єднання, переможцем є один з l нейронів шару Кохонена. В інших випадках відбувається некоректна класифікація.

Алгоритм навчання нейромережевого детектора [55, 59, 60]

представлений на рисунку 2.6.

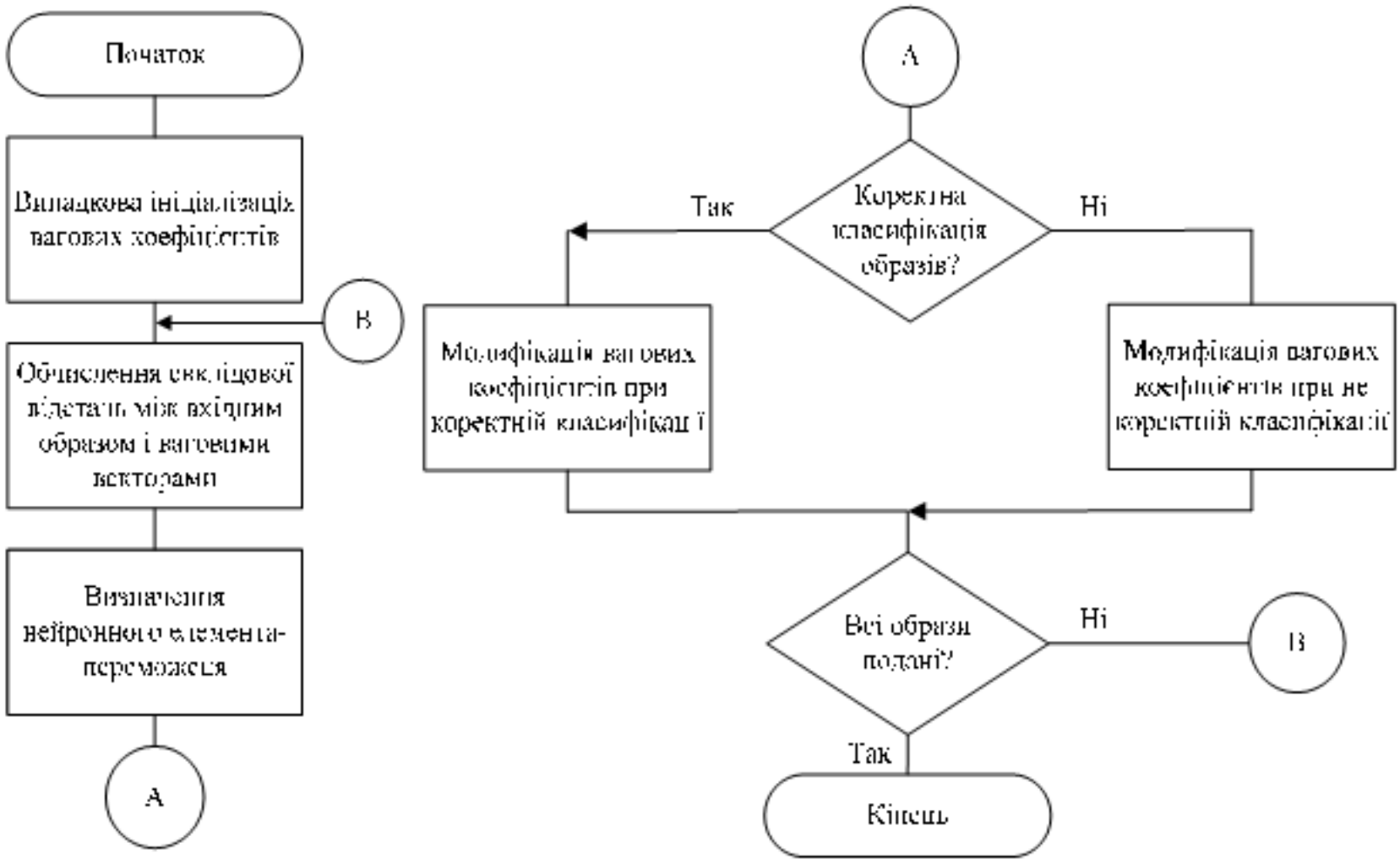


Рисунок 2.6 – Алгоритм навчання нейромережевого детектора

Таким чином, алгоритм навчання LVQ мережі можна представити у вигляді послідовності наступних кроків [51, 59]:

1. Випадкова ініціалізація вагових коефіцієнтів ω_{ci} нейронів Y_i шару Кохонена.

2. Розподіл вхідного образу (вектор, що складається з 41 значення параметрів мережевого з'єднання) з навчальної вибірки на нейронну мережу і обчислення наступних параметрів:

a) евклідова відстань між вхідним образом і ваговими векторами нейронних елементів шару Кохонена за формулою (2.6).

b) визначається нейронний елемент-переможець з номером k

$$D_k = \min_i D_i. \qquad (2.12)$$

с) проводиться модифікація вагових коефіцієнтів нейрона-переможця. Причому, у разі коректної класифікації, тобто, якщо при подачі на вхід нейронної мережі параметрів мережевої атаки переможцем є один з f нейронів, або, якщо при подачі на вхід нейронної мережі параметрів нормального мережевого з'єднання переможцем є один з l нейронів шару Кохонена, то модифікація вагових коефіцієнтів проводиться згідно наступного виразу:

$$\omega_{ck}(t+1) = \omega_{ck}(t) + \gamma(X_c - \omega_{ck}(t)), \tag{2.13}$$

де γ – крок навчання.

Інакше, тобто при некоректній класифікації, вагові коефіцієнти нейронів шару Кохонена модифікуються згідно виразу:

$$\omega_{ck}(t+1) = \omega_{ck}(t) - \gamma(X_c - \omega_{ck}(t)). \tag{2.14}$$

3. Процес повторюється, починаючи з пункту 2, для всіх вхідних образів.

4. Навчання проводиться до бажаного ступеня узгодження між вхідними і ваговими векторами, тобто до тих пір, поки значення сумарної квадратичної помилки мережі не стане мінімальним.

Як вже наголошувалося, вагові коефіцієнти нейронів третього шару (див. рисунок 2.5), які характеризують зв'язки між f нейронами шару Кохонена і першим нейроном останнього шару, а також між l нейронами шару Кохонена і другим нейроном останнього шару, мають одиничні значення. Решта вагових коефіцієнтів нейронів останнього шару дорівнює нулю.

У даному параграфі представлений алгоритм навчання нейромережевого детектора для виявлення мережевих атак, який базується на контрольованому конкурентному навчанні і характеризується тим, що в

процесі навчання коректна класифікація вхідного образу відбувається, якщо при подачі на вхід нейронної мережі параметрів з'єднання, що відносяться до класу мережевої атаки або нормального з'єднання, переможцем є відповідно один з *f* або *l* нейронів шару Кохонена.

2.3.3 Структура нейронів прихованого шару і навчальної вибірки

Розглянемо вибір структури нейронів прихованого шару, яка визначається співвідношенням кількості нейронів в шарі Кохонена, що здійснюють кластеризацію атак, і нормальних з'єднань (*f*/*l*). Відповідно до цього визначається структура навчальної вибірки нейромережевого детектора, тобто відношення кількості атак до кількості нормальних з'єднань в навчальній вибірці.

Для вибору структури нейронів прихованого шару і навчальної вибірки проведемо ряд експериментів, умови проведення яких представлені в таблиці 2.7.

Таблиця 2.7 – Умови проведення експериментів

Навчальна вибірка	80 векторів	
Адаптація параметрів з'єднання	параметр protocol_type: tcp – 1; udp – 2; icmp – 3. параметр service: http – 1; smtp – 2; …; telnet – 5 .	
Структура нейромережевого детектора	41-10-2	
Тестова вибірка	DOS	391458
	Probe	4107
	R2L	1126
	U2R	52
	Normal	97278

У таблиці 2.8 приведені результати експериментів нейромережевих детекторів з різним співвідношенням кількості атак до кількості нормальних з'єднань в навчальній вибірці і відповідно різним співвідношенням кількості нейронів в шарі Кохонена.

Таблиця 2.8 – Достовірність виявлення атак

Клас атак	5/1	4/1	3/1	2/1	1/1
DOS	95,7%	98,0%	97,5%	96,4%	96,0%
Probe	59,2%	65,1%	63,9%	62,1%	61,5%
R2L	32,4%	36,9%	34,8%	33,9%	33,0%
U2R	16,8%	20,8%	19,0%	18,7%	17,1%

Було проведено п’ять експериментів. У першому експерименті для навчання нейронної мережі використовувалася така навчальна вибірка, в якій співвідношення нормальних мережевих з’єднань до атак складало п’ять до одного. У другому експерименті співвідношення класів мережевих з’єднань дорівнювало чотири до одного, в третьому – три до одного, в четвертому – два до одного. В останньому експерименті навчальна вибірка складалася з 50% нормального трафіку і 50% мережевих атак. Згенеровані детектори навчалися і класифікували невідомі образи.

Як видно з таблиці 2.8, якнайкращий результат показали ті детектори, для навчання яких використовувалася вибірка, що складається з 80% мережевих атак і 20% нормальних мережевих з’єднань, тобто співвідношення атак до нормальних з’єднань дорівнює чотири до одного [62].

Так як класи мережевого трафіку розподілені в навчальній вибірці в співвідношенні 80% мережевих атак і 20% нормальних з’єднань, то це накладає певні вимоги на розподіл нейронів Кохонена в прихованому шарі нейромережевого детектора. Для коректного функціонування вибраної нейронної мережі необхідно, щоб співвідношення між кількістю нейронів в шарі Кохонена, що характеризують різні класи, повинно бути кратним співвідношенню чотири до одного.

Таким чином, співвідношення нейронів в прихованому шарі повинне бути рівним [62]:

$$f/l = 4/1 \tag{2.15}$$

де f – перші нейрони шару Кохонена, активність яких характеризує

мережеву атаку;

l – останні нейрони шару Кохонена, активність яких характеризує нормальне мережеве з'єднання.

В результаті, якщо кількість нейронів Кохонена в прихованому шарі дорівнює десяти, то кількість нейронів, що відповідають за мережеві атаки, буде рівним $f = 8$, а кількість нейронів, що відповідають за нормальні з'єднання, буде рівним $l = 2$.

Для підвищення здатності нейромережевого детектора виявляти атаки, як буде показано в наступному розділі, необхідно застосовувати стиснення вхідного простору даних для отримання найбільш інформативних вхідних ознак.

У даному параграфі представлений метод побудови нейромережевих детекторів для виявлення комп'ютерних атак, який базується на нейронній мережі векторного квантування LVQ, де 80% нейронних елементів у схованому шарі відповідають типу атаки, а решта – нормальному з'єднанню. Запропоновано структуру навчальної вибірки нейромережевого детектора, співвідношення атак і нормальних з'єднань в якій становить 4:1. Запропонований метод характеризується малим обсягом навчальної вибірки і дозволяє більш точно розділити один клас мережевих з'єднань від іншого з метою підвищення достовірності виявлення атак на ІТМ.

2.4 Метод побудови сукупного класифікатора для ієрархічної класифікації атак

2.4.1 Стиснення вхідних даних на основі методу головних компонент

Як вже зазначалось, виділяють 41 параметр мережевого з'єднання, які поступають на вхід нейромережевого детектора для його навчання і аналізу трафіку, з метою виявлення мережевих атак. Проведені експерименти показують здатність розроблених детекторів виявляти мережеві атаки. Проте, частина атак, таких як *loadmodule*, *perl*, *spy*, *guess_passwd*, залишаються

такими, що практично не детектуються. Багато в чому це виникає унаслідок того, що використовувана для тесту база з'єднань KDD-99 містить недостатню кількість записів про ці атаки і, відповідно, дані, на яких можна навчити нейронну мережу.

Для того, щоб поліпшити якість навчання нейронних мереж на «рідкісних» типах атак, пропонується використовувати метод головних компонент [190, 191] для скорочення розміру інформації при навчанні та аналізі мережевого трафіку [52, 55, 60, 62].

Метод головних компонент (principal component analysis, PCA) є одним з основних способів зменшення розмірності даних при мінімальній втраті інформації. В даний час метод застосовується для вирішення багатьох інженерних задач, таких як, стиснення даних і т. д. Обчислення головних компонент зводиться до обчислення власних векторів і власних значень коваріційної матриці початкових даних.

Метод головних компонент полягає в лінійному ортогональному перетворенні вхідного вектора X розмірності n у вихідний вектор Y розмірності p, де $p<n$. Сукупність вхідних образів представимо у вигляді матриці:

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \dots & \dots & \dots & \dots \\ x_1^L & x_2^L & \dots & x_n^L \end{bmatrix}, \tag{2.16}$$

де $X^k = \left(x_1^k, x_2^k \dots x_n^k\right)$ відповідає k-му вхідному образу;

L - загальна кількість образів.

Вважатимемо, що матриця X є центрованою, тобто вектор математичних очікувань $\mu=0$. Матриця коваріацій вхідних даних X визначається як

$$K = E\{XX^T\} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}, \qquad (2.17)$$

де σ_{ij} - коваріація між i-ою і j-ою компонентою вхідних образів.

Метод головних компонент полягає у знаходженні таких лінійних комбінацій початкових змінних X, що вихідні змінні Y (головні компоненти) некорельовані між собою, впорядковані за збільшенням дисперсії і сума дисперсій вхідних образів (у разі $n=p$) після перетворення залишається без змін. Тоді підмножина перших p головних компонент характеризує велику частину загальної дисперсії. В результаті виходить стисле представлення вхідної інформації.

У загальному вигляді метод головних компонент можна представити як:

$$Y = XP, \qquad (2.18)$$

де P – ортонормована матриця лінійного перетворення розмірності $n \times p$, стовпці якої відповідають власним векторам коваріаційної матриці.

Таке перетворення можна представити в наступному вигляді:

$$KP = P\lambda\,, \qquad (2.19)$$

або

$$P^T KP = \lambda \qquad (2.20)$$

де λ – діагональна матриця власних значень коваріаційної матриці K.

Також слід зазначити, що метод головних компонент можна реалізувати за допомогою рециркуляційної нейронної мережі, яка навчається за правилом Ойя [186].

На рисунках 2.7–2.10 представлений розподіл в тривимірному просторі перших головних компонент різних мережевих атак і нормальних з'єднань, що візуально демонструє взаємозв'язок між даними, які розглядаються [55, 60], а в додатку К представлений розподіл всіх мережевих атак і нормальних з'єднань.

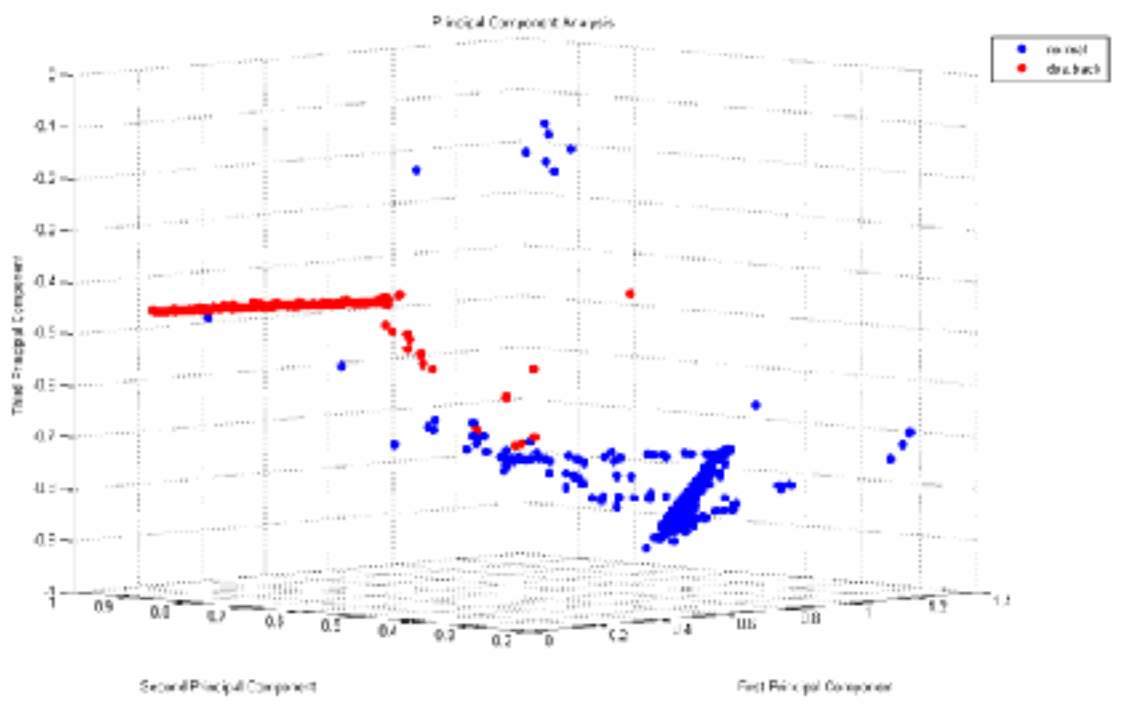


Рисунок 2.7 – Розподіл мережевої атаки *dos_back* і нормальних з'єднань

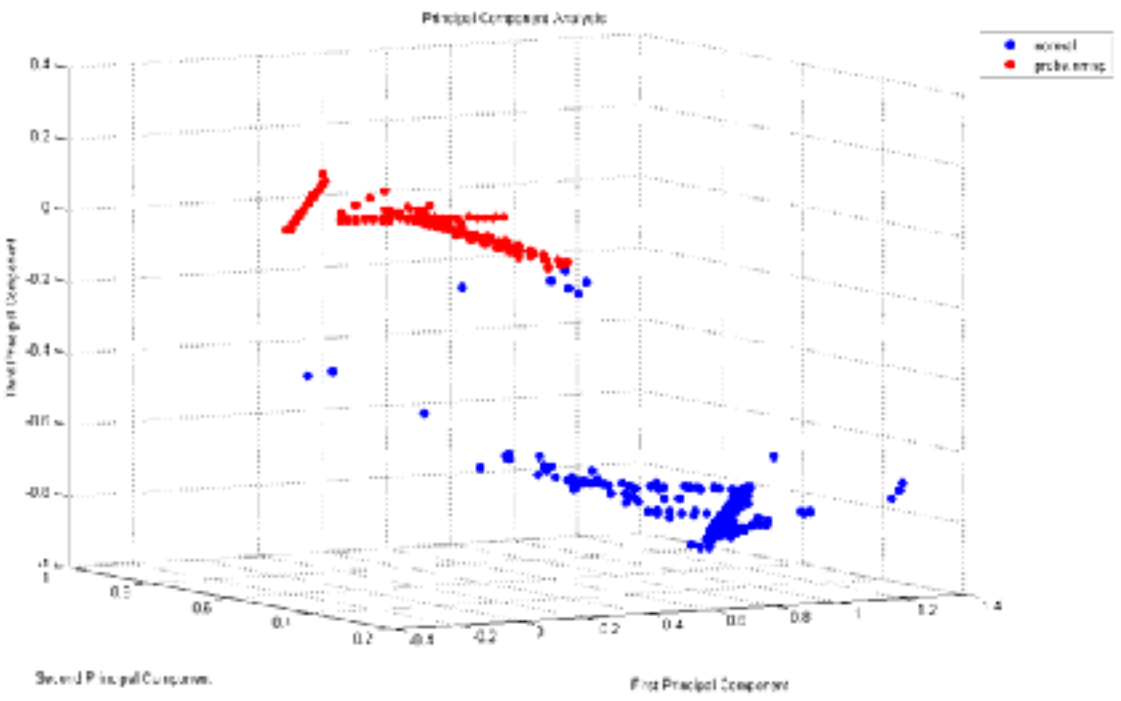


Рисунок 2.8 – Розподіл мережевої атаки *probe_nmap* і нормальних з'єднань

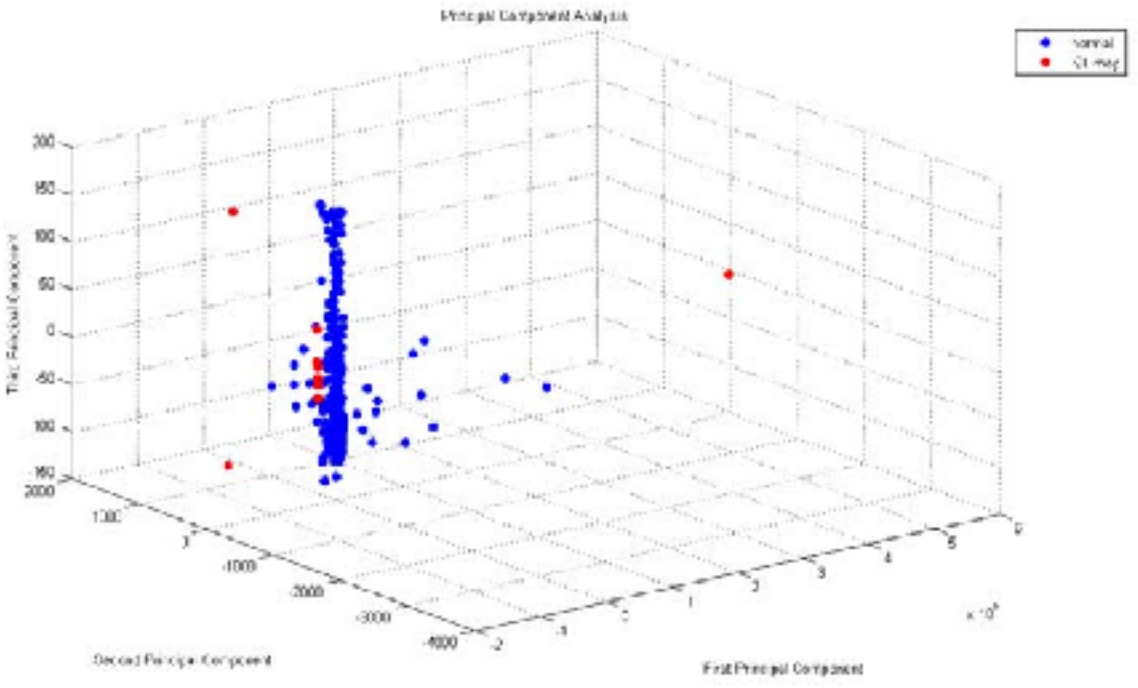


Рисунок 2.9 – Розподіл мережевої атаки *r2l_imap* і нормальних з’єднань

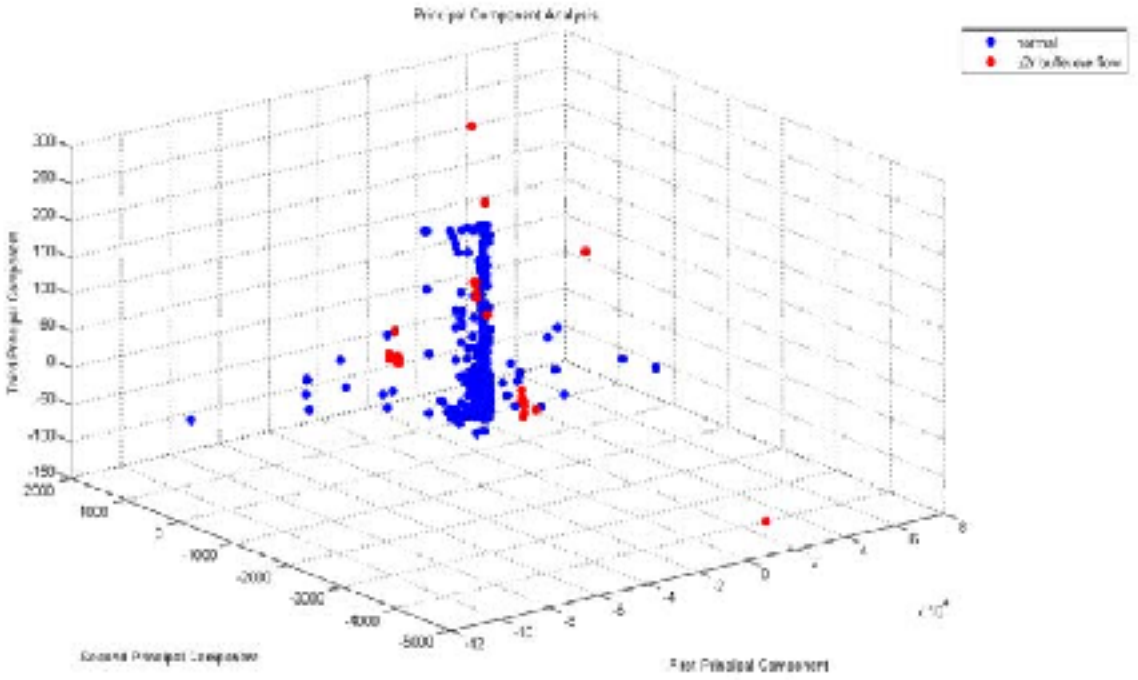


Рисунок 2.10 – Розподіл мережевої атаки *u2r_bufferoverflow* і нормальних з’єднань

Як видно з рисунків, даний розподіл має нелінійний характер. Визначимо число головних компонент, які використовуватимемо для аналізу мережевого трафіку. Для цього розглянемо наступний критерій відносної інформативності [145]:

$$J = \frac{\lambda_1 + \lambda_2 + \ldots + \lambda_p}{\lambda_1 + \lambda_2 + \ldots + \lambda_n} \qquad (2.21)$$

Аналізуючи за допомогою виразу (2.21) розподіл кількості інформації, що міститься в кожній подальшій головній компоненті, можна визначити число головних компонент *p*, які доцільно використовувати для подальшого аналізу без істотної втрати інформативності *J*. Застосування методу головних компонент для зменшення розмірності даних, що описують мережевий трафік (як вже відомо, мережевий трафік описується за допомогою 41 параметра), показало, що в 11 головних компонентах міститься 99% інформації про мережеве з'єднання. Розподіл інформативності про мережеве з'єднання залежно від компоненти представлено в таблиці 2.9 [55, 60].

Таблиця 2.9 – Розподіл інформації в головних компонентах

Номер компоненти	1	2	3	4	5	6	7	8	9
Кількість інформації, %	52,40	71,67	88,37	91,49	94,21	95,90	96,96	97,71	98,27
Номер компоненти	10	11	12	13	14	15	16	17	18
Кількість інформації, %	98,73	99,00	99,18	99,33	99,47	99,59	99,67	99,75	99,81
Номер компоненти	19	20	21	22	23	24	25	26	27
Кількість інформації, %	99,87	99,90	99,93	99,94	99,95	99,96	99,97	99,98	99,98
Номер компоненти	28	29	30	31	32	33	34	35	36
Кількість інформації, %	99,99	99,99	99,99	99,99	99,99	99,99	99,99	99,99	99,99
Номер компоненти	37	38	39	40	41				
Кількість інформації, %	99,99	100	100	100	100				

Таблиця організована таким чином, що показує підсумовувану

кількість інформації залежно від кількості компонент. Тобто, одна головна компонента містить 52,40% інформації, дві головних компоненти містять вже 71,67% інформації, три – 88,37% і т. д.

На рисунку 2.11 представлена залежність кількості інформації від числа головних компонент [62].

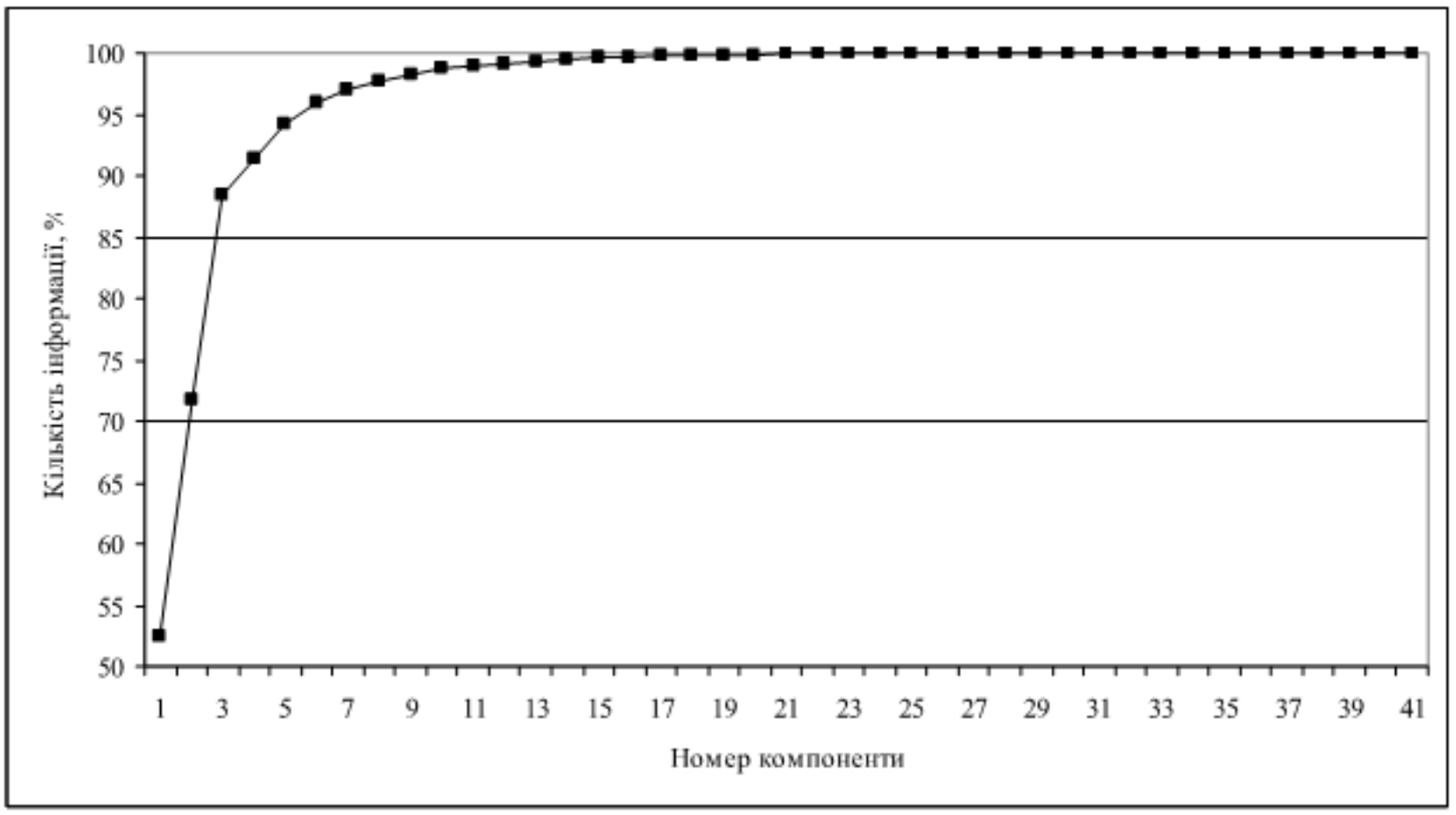


Рисунок 2.11 – Залежність кількості інформації від числа головних компонент

Як видно з таблиці 2.9 і рисунку 2.11, перших 12 головних компонент містять більше 99% інформації про мережевий трафік. У останніх 30 компонентах міститься менше 1% інформації, і, з міркування доцільності, їх можна виключити з аналізу.

Виходячи з того, що 41 параметр, які описують мережевий трафік, за своєю структурою надмірні, і що для опису мережевого трафіка достатньо 11–20 головних компонент, в яких зберігається 99,00–99,90 відсотків інформації, доцільно застосовувати дане перетворення даних перед тим, як їх аналізувати за допомогою нейронних мереж.

Результати експериментальних досліджень показали, що при використанні методу головних компонент для попередньої обробки

інформації про мережеві з'єднання, достовірність виявлення атак на ІТМ підвищується. Експерименти проводилися на базі атак *DoS* і *Probe* відповідно до умов представлених в таблиці 2.7. Структура нейронної мережі у випадку використання PCA – 12-10-2 (12 вхідних нейронів, 10 нейронів прихованого шару і 2 вихідних нейрона). Отримані результати представлені в таблицях 2.10 і 2.11.

Таблиця 2.10 – Порівняльний аналіз результатів виявлення DoS-атак

	Back, %	Land, %	Neptune, %	Pod, %	Smurf, %	Teardrop, %
PCA	99,5	100,0	100,0	98,1	100,0	100,0
без PCA	99,5	90,5	100,0	98,1	100,0	100,0
Покращення	0	9,5	0,0	0,0	0,0	0,0

Таблиця 2.11 – Порівняльний аналіз результатів виявлення Probe-атак

	Ipsweep, %	Nmap, %	Portsweep, %	Satan, %
PCA	65,2	100,0	99,9	99,3
без PCA	7,1	54,5	99,6	99,3
Покращення	58,1	45,5	0,3	0,0

Таким чином, для успішного аналізу мережевого трафіку досить використовувати 12 перших головних компонент, в яких міститься більше 99% інформації про мережеве з'єднання, а не 41 параметр. Це дозволяє істотно прискорити як процес навчання нейромережевого детектора, так і процес аналізу мережевого трафіку. Для цього, до виділених з мережевого трафіку даних необхідно застосувати спочатку метод головних компонент, а потім подати отримані дані на вхід нейронної мережі.

Отже, застосування методу головних компонент дозволило зменшити розмірність аналізованої інформації в 3,4 рази при втраті інформативності 0,8%.

2.4.2 Сукупний класифікатор для ієрархічної класифікації атак

Сукупним нейромережевим детектором є такий детектор, який складається з множини нейромережевих детекторів, кожен з яких навчений на певному типі атак, що дозволяє детектувати клас і тип атаки. Розглянемо різні підходи до побудови сукупного класифікатора.

У підрозділі 2.3 представлений метод побудови нейромережевого детектора для виявлення атак на ІТМ. Такий детектор складається з трьохшарової нейронної мережі, де в якості прихованого шару використовується шар Кохонена. Детектор навчається на навчальній вибірці, що складається з параметрів мережевого трафіку, причому 20% навчальної вибірки складають нормальні мережеві з'єднання, а 80% - один з типів атак.

Оскільки виділяють 22 різновиди мережевих атак (див. додаток А), які об'єднані в чотири класи (*DoS*, *U2R*, *R2L* і *Probe*) [81], то для побудови сукупного класифікатора використовуються 22 нейромережевих детектори, які навчені на виявлення кожного з 22 різновидів мережевих атак.

Схему функціонування сукупного нейромережевого детектора для виявлення і класифікації мережевих атак можна представити у вигляді рисунку 2.12 [48–50].

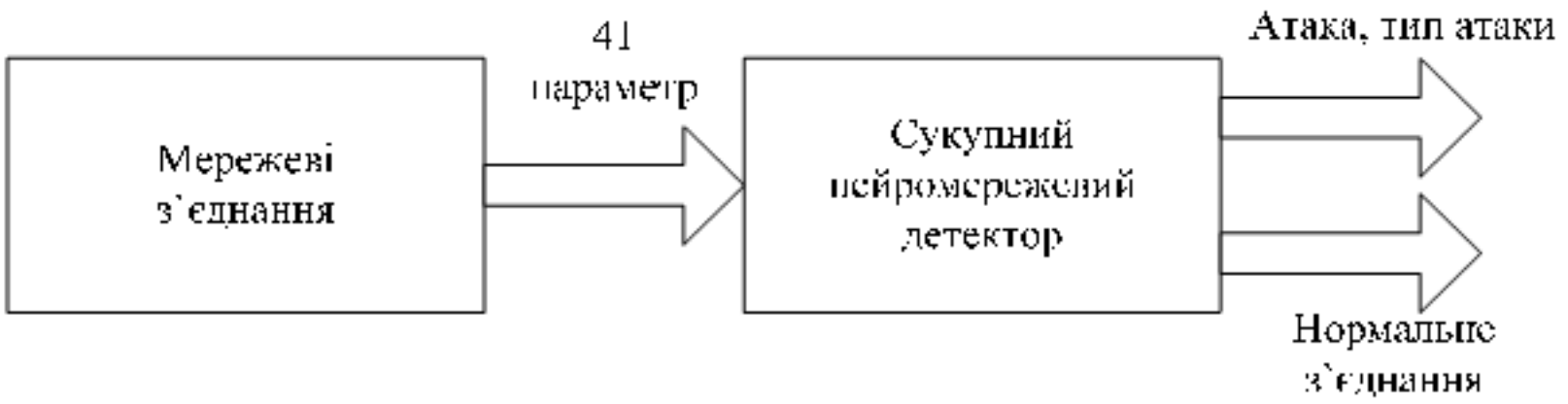


Рисунок 2.12 – Схема функціонування сукупного нейромережевого детектора

Значення параметрів мережевих з'єднань подаються на сукупний нейромережевий детектор. Активність одного з детекторів вказує на виявлення аномального з'єднання, тип якого відповідає тій мережевій атаці, на виявлення якої навчався даний детектор. Якщо жоден з детекторів не

реагує на дане з'єднання, то воно вважається нормальним.

Таким чином, алгоритм функціонування детекторів виявлення і класифікації мережевих атак на основі сукупності нейромережевих детекторів можна представити в наступному вигляді (рисунок 2.13) [51, 59]:

1. Отримання параметрів із встановленого мережевого з'єднання.

2. Перетворення параметрів мережевого з'єднання у відповідні числові значення.

3. Подача параметрів мережевого з'єднання на *j*-й нейромережевий детектор.

4. Якщо вихідне значення *j*-го нейромережевого детектора дорівнює $Y_j=1$, то мережеве з'єднання вважається мережевою атакою і блокується. Користувачеві видається повідомлення про виявлення типу атаки, яка відповідає номеру детектора. Якщо вихідне значення нейромережевого детектора дорівнює $Y_j=0$, то параметри з'єднання подаються на наступний детектор.

5. Кроки 3 і 4 повторюються до тих пір, поки всі 22 детектори не перевірять мережевий трафік. Якщо вихідні значення всіх детекторів дорівнюють нулю, то з'єднання вважається нормальним.

Основним недоліком такого підходу до побудови сукупного класифікатора є можливість наявності конфліктів в роботі детекторів, коли декілька різних детекторів сигналізують про атаку. В цьому випадку неможливо точно класифікувати атаку.

Також, як вже наголошувалося в параграфі 2.4.1, доцільно застосовувати метод головних компонент для скорочення розмірності вхідного простору даних, що дозволяє скоротити час навчання детекторів. З урахуванням цього, узагальнену схему функціонування сукупного нейромережевого детектора для виявлення мережевих атак з використанням методу головних компонент можна представити таким чином (рисунок 2.14) [62].

Рисунок 2.13 – Узагальнений алгоритм функціонування сукупного нейромережевого детектора виявлення і класифікації атак на ІТМ

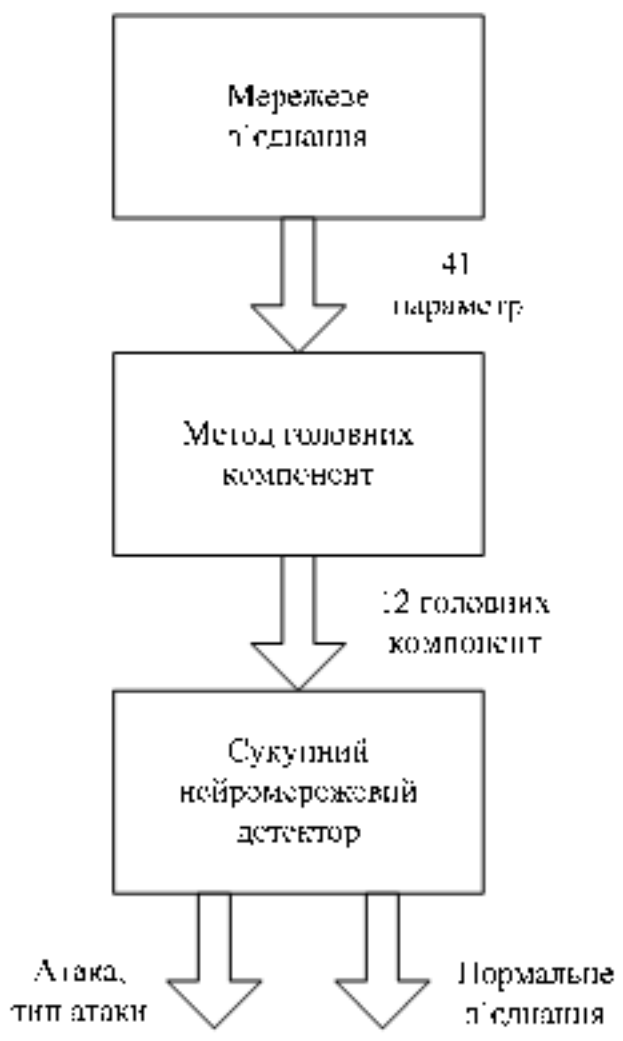


Рисунок 2.14 – Взаємодія PCA і сукупного нейромережевого детектора

При такому підході, кількість *n* вхідних нейронів нейронної мережі, що використовується в якості детектора, дорівнює 12. Вхідною інформацією є 12 перших головних компонент, які подаються на 22 нейромережеві детектори, де і відбувається її визначення до класу мережевої атаки або до класу нормального з'єднання.

Схема сукупного класифікатора для ієрархічної класифікації атак на ІТМ представлена на рисунку 2.15 [57, 64].

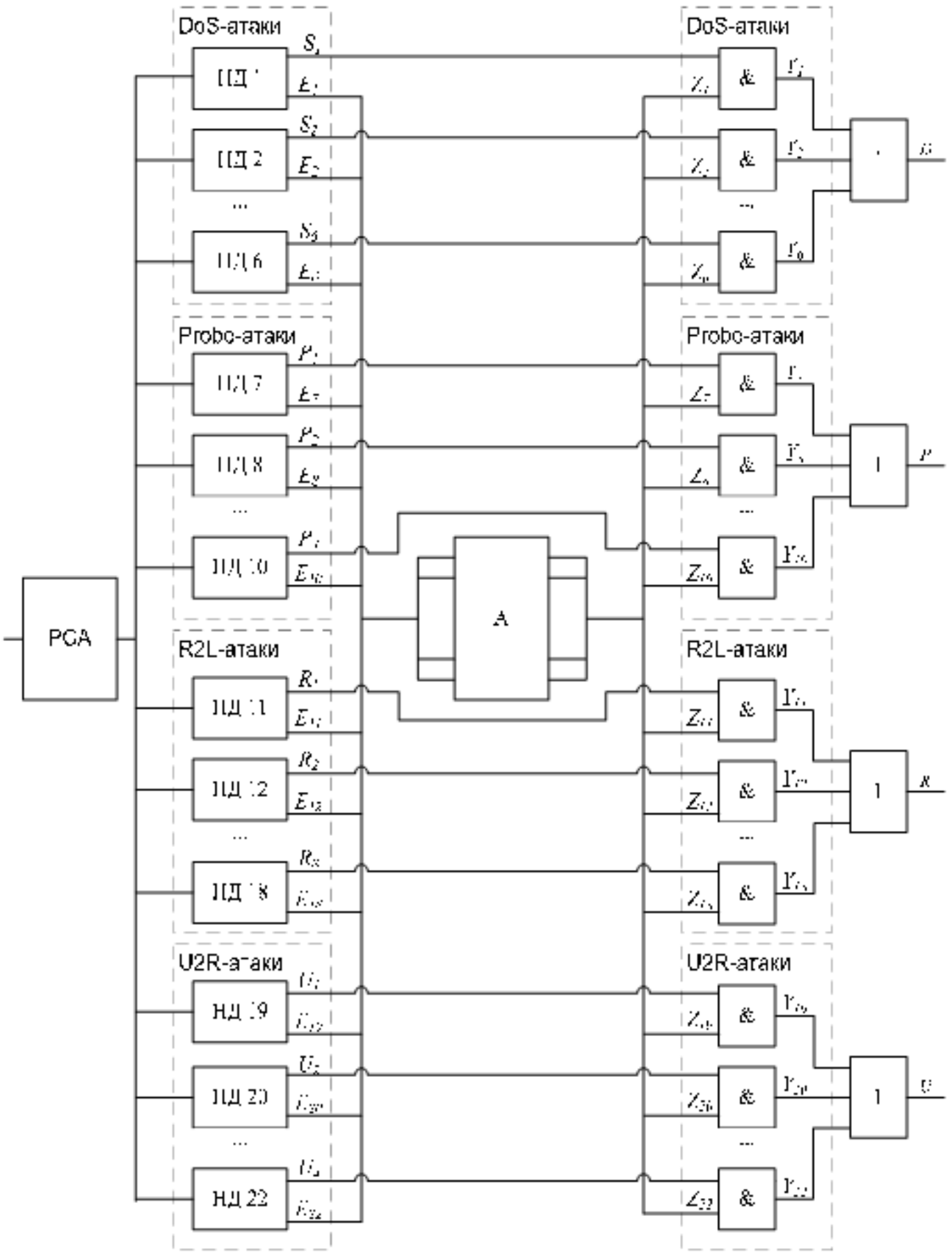


Рисунок 2.15 – Схема сукупного класифікатора для ієрархічної класифікації атак на ІТМ

На рисунку 2.15 представлений приклад схеми побудови сукупного класифікатора для ієрархічної класифікації атак, який базується на стисненні інформації з використанням методу головних компонент і на такому об'єднанні нейромережевих детекторів, щоб нейтралізувати конфлікти між ними. Для нейтралізації конфліктів між детекторами використовується евклідова відстань між вхідним образом і ваговими векторами нейронів-переможців кожного з детекторів. Детектор, який має мінімальну евклідову відстань, є переможцем в конкурентній боротьбі і визначає клас і тип атаки.

Розглянемо функціонування сукупного класифікатора. Стиснений набір вхідних даних розмірністю 12 поступає на нейромережеві детектори, кожен з яких навчений на відповідний тип атак. В результаті, якщо детектор виявляє атаку, то вихідне значення його першого виходу встановлюється в одиничне значення. Для усунення конфліктів в роботі такого класифікатора, коли декілька детекторів встановлюються в одиничний стан, на другий вихід кожного детектора передається мінімальна евклідова відстань між вхідним образом і ваговими векторами відповідного детектора:

$$E_j = \min_j D_j = \min_i \sqrt{(x_1 - w_{1j})^2 + (x_2 - w_{2j})^2 + ... + (x_{12} - w_{12j})^2} \,. \quad (2.22)$$

Інформація про мінімальну евклідову відстань поступає з кожного детектора на арбітр, який визначає детектор з номером k, що має мінімальну евклідову метрику:

$$E_k = \min E_j,\ j = \overline{1,22} \,. \quad (2.23)$$

В результаті k-й вихід арбітра встановлюється в одиничний стан, а решта виходів – в нульовий стан:

$$Z_i = \begin{cases} 1, & якщо\ i = k \\ 0, & інакше \end{cases}. \qquad (2.24)$$

На виходах логічних елементів «І» визначається тип атаки:

$$Y_i = F_i Z_i, \qquad (2.25)$$

де $F_i = \begin{cases} S_i, якщо\ i = \overline{1,6} \\ P_i, якщо\ i = \overline{7,10} \\ R_i, якщо\ i = \overline{11,18} \\ U_i, якщо\ i = \overline{19,22} \end{cases}$

Виходи логічних елементів «АБО» визначають клас атаки:

$$D = \bigvee_{i=1}^{6} Y_i, \quad P = \bigvee_{i=7}^{10} Y_i, \quad R = \bigvee_{i=11}^{18} Y_i, \quad U = \bigvee_{i=19}^{22} Y_i, \qquad (2.26)$$

де D – *DoS*-атака;

P– *Probe*-атака;

R– *R2L*-атака;

U– *U2R*-атака.

У даному параграфі запропоновано ієрархічний класифікатор атак на ІТМ, який складається із сукупності багатоканальних нейромережевих детекторів, кожний з яких навчений на певний тип атаки і дозволяє визначати тип і клас мережевих атак. Вхідною інформацією сукупного класифікатора є 12 перших головних компонент, які подаються на 22 нейромережевих детектори. Запропонована схема усунення конфліктів в роботі нейромережевих детекторів.

Висновки до розділу 2

1. Розроблено узагальнену функціональну модель обробки інформації при виявленні і класифікації атак на ІТМ, яка складається з трьох етапів:

захоплення параметрів мережевого з'єднання, обробка інформації, нейромережевий аналізатор, що дозволило підвищити достовірність виявлення і класифікації атак на ІТМ.

2. Проведено вибір нейронної мережі для виявлення і класифікації атак на ІТМ. В якості основи нейромережевого детектора вибрана нейронна мережа векторного квантування LVQ з нейронними елементами Кохонена в прихованому шарі, що дозволило мінімізувати розмірність навчальної вибірки (>= 20 векторів) і, відповідно, час навчання.

3. Вдосконалено метод побудови нейромережевих детекторів атак, який базується на нейронній мережі векторного квантування LVQ, де 80% нейронних елементів Кохонена в прихованому шарі відповідають типу атаки, а решта – нормальному з'єднанню. Відповідно, запропонована структура навчальної вибірки, співвідношення атак і нормальних з'єднань в якій дорівнює 4:1. Запропонований метод характеризується малим обсягом навчальної вибірки, що дозволило окремо здійснити кластеризацію атак і нормальних з'єднань в прихованому шарі в порівнянні з іншими підходами з метою підвищення достовірності виявлення і класифікації атак на ІТМ.

4. Запропоновано алгоритм навчання нейромережевого детектора, який характеризується тим, що в процесі навчання коректна класифікація відбувається у тому випадку, коли при подачі на вхід нейронної мережі параметрів з'єднання, що відноситься до класу мережевої атаки, переможцем є один з перших нейронів шару Кохонена, або якщо при подачі на вхід нейронної мережі параметрів нормального з'єднання переможцем є один з останніх нейронів шару Кохонена.

5. Вдосконалено метод побудови сукупного класифікатора для ієрархічної класифікації атак на ІТМ на основі багатоканальних нейромережевих детекторів, що дало можливість зменшити розмірність аналізованої інформації в 3,4 рази при втраті інформативності 0,8% за рахунок стиснення вхідної інформації для отримання найбільш інформативних ознак, а також класифікувати типи і класи атак за рахунок

об’єднання навчених на певний тип атаки нейромережевих детекторів. Запропонований метод дозволив усунути конфлікти в роботі нейромережевих детекторів.

РОЗДІЛ 3
КОМБІНОВАНИЙ МЕТОД ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ

3.1 Основи технології виявлення і класифікації атак з використанням імунних систем

У даному розділі представлена інтеграція методів штучних імунних систем і штучних нейронних мереж для підвищення достовірності виявлення і класифікації атак на ІТМ. В якості базового принципу побудови комбінованого методу використовується механізм штучних імунних систем, де в якості імунного детектора використовується нейронна мережа. Використання штучних імунних систем разом з штучними нейронними мережами для вирішення поставленої задачі дозволило підвищити достовірність виявлення і класифікації атак на ІТМ, а також зробити систему захисту гнучкішою і здатною донавчатися, що дозволило їй адаптуватися до виявлення нових, раніше невідомих типів атак [47, 54, 58].

Побудова комбінованого методу здійснюється на основі базових принципів і механізмів біологічної імунної системи, опис яких представлений в першому розділі, а також на типовій схемі штучної імунної системи, запропонованої в [192]. Це такі механізми, як генерація і навчання імунних детекторів, відбір детекторів, які з певних причин генерують помилкові рішення, функціонування детекторів, активація детекторів і формування імунної пам’яті (рисунок 3.1) [54, 58].

Імунні детектори генеруються по випадковому алгоритму, що дає можливість створення великої кількості різноманітних за своєю структурою детекторів, які здатні реагувати на будь-яку аномалію. Потім, детектори проходять стадію навчання, на якій вони набувають здатності коректно реагувати на чужорідні об’єкти або явища. Для того, щоб детектори не генерували помилкові спрацьовування, вони ретельно відбираються. Ті з них,

які не навчилися коректно класифікувати об’єкти, – знищуються.

Рисунок 3.1 – Схема процесу функціонування детекторів штучної імунної системи

Відібрані детектори допускаються до виконання функцій по класифікації об’єктів. Кожному детектору надається деяка лімітована кількість часу (час життя), впродовж якого він може існувати. Якщо впродовж цього часу детектор не виявляє аномалій, то він знищується, а на його місце приходить новий детектор. Якщо детектор виявив аномалію, відбувається, так звана, стадія активації. На цій стадії відбувається інформування про виявлену аномалію і її знищення. Детектор, що виявив аномалію, трансформується в детектор імунної пам’яті. Детектори імунної пам’яті характеризуються великим часом життя і рівнем довіри.

Штучна імунна система моделює основні процеси біологічної імунної системи, а також їх взаємодію. Відмінність полягає в способі представлення інформації і структурі імунного детектора.

У роботі запропонований комбінований метод виявлення і класифікації атак на ІТМ, що базується на інтеграції штучних імунних і нейронних мереж, що дало можливість підвищити достовірність виявлення і класифікації мережевих атак. Розглянемо його основні положення.

3.2 Розробка комбінованого методу виявлення і класифікації атак на інформаційні телекомунікаційні мережі

В розділі 2 представлені метод побудови нейромережевого детектора для виявлення і класифікації атак на ІТМ і метод побудови сукупного класифікатора для ієрархічної класифікації мережевих атак. У даному розділі пропонується метод виявлення і класифікації атак, що базується на технології штучних імунних систем, де в якості імунних детекторів використовуються нейронні мережі. Основна ідея полягає у використанні базових принципів функціонування штучної імунної системи, таких як навчання і відбір імунних детекторів, виявлення і класифікація атак множиною імунних детекторів, клонування і мутація детекторів, формування імунної пам'яті з кращих імунних детекторів.

В якості імунного детектора використовується розроблений в другому розділі нейромережевий детектор, в основі якого лежить нейронна мережа векторного квантування (LVQ). Такий нейромережевий детектор характеризується розділенням нейронів в прихованому шарі Кохонена, що відповідають за різні класи – мережеві атаки і нормальні з'єднання, малим об'ємом навчальної вибірки, що дозволяє оперативно генерувати різноманітні імунні детектори для виявлення і класифікації різних типів атак.

Поставимо у відповідність кожному класу атак свій набір нейромережевих імунних детекторів, які навчаються виявляти певні типи

атак в кожному класі. При цьому, в загальному випадку, декілька детекторів можуть навчатися виявляти один і той же тип мережевої атаки.

Тоді узагальнений метод виявлення і класифікації атак на ІТМ на основі штучних імунних систем, де в якості імунних детекторів використовуються нейронні мережі, складатиметься з наступних кроків [54]:

1. *Генерація імунних детекторів.* На даному кроці відбувається генерація чотирьох груп нейромережевих детекторів відповідно до кількості класів атак з випадковою ініціалізацією вагових коефіцієнтів, де кожна група складається з множини детекторів:

$$\begin{aligned} D &= \left\{ D_i, \quad i = \overline{1, F_d} \right\}, \\ P &= \left\{ P_i, \quad i = \overline{1, F_p} \right\}, \\ R &= \left\{ R_i, \quad i = \overline{1, F_r} \right\}, \\ U &= \left\{ U_i, \quad i = \overline{1, F_u} \right\}, \end{aligned} \tag{3.1}$$

де D_i – i-й нейромережевий імунний детектор (НІД) для виявлення і класифікації *DoS*-атак;

P_i – i-й НІД для виявлення і класифікації *Probe*-атак;

R_i – i-й НІД для виявлення і класифікації *R2L*-атак;

U_i – i-й НІД для виявлення і класифікації *U2R*-атак;

F – загальна кількість детекторів відповідного типу.

Також, як і в біологічному імунітеті, штучні імунні детектори повинні пройти процес навчання, для того, щоб коректно виконувати задачу виявлення і класифікації атак на ІТМ.

2. *Навчання імунних детекторів.* На даному етапі згенеровані нейромережеві детектори піддаються процесу навчання. Як було сказано раніше, для навчання нейромережевих детекторів використовується контрольоване конкурентне навчання [23] відповідно до правила «переможець бере все». Проте механізм навчання імунного детектора дещо відрізняється від того, який був представлений раніше (див. розділ 2). Розглянемо докладніше пропонований механізм навчання для імунних

детекторів (рисунок 3.2), в основі яких лежить вибрана нейронна мережа [54].

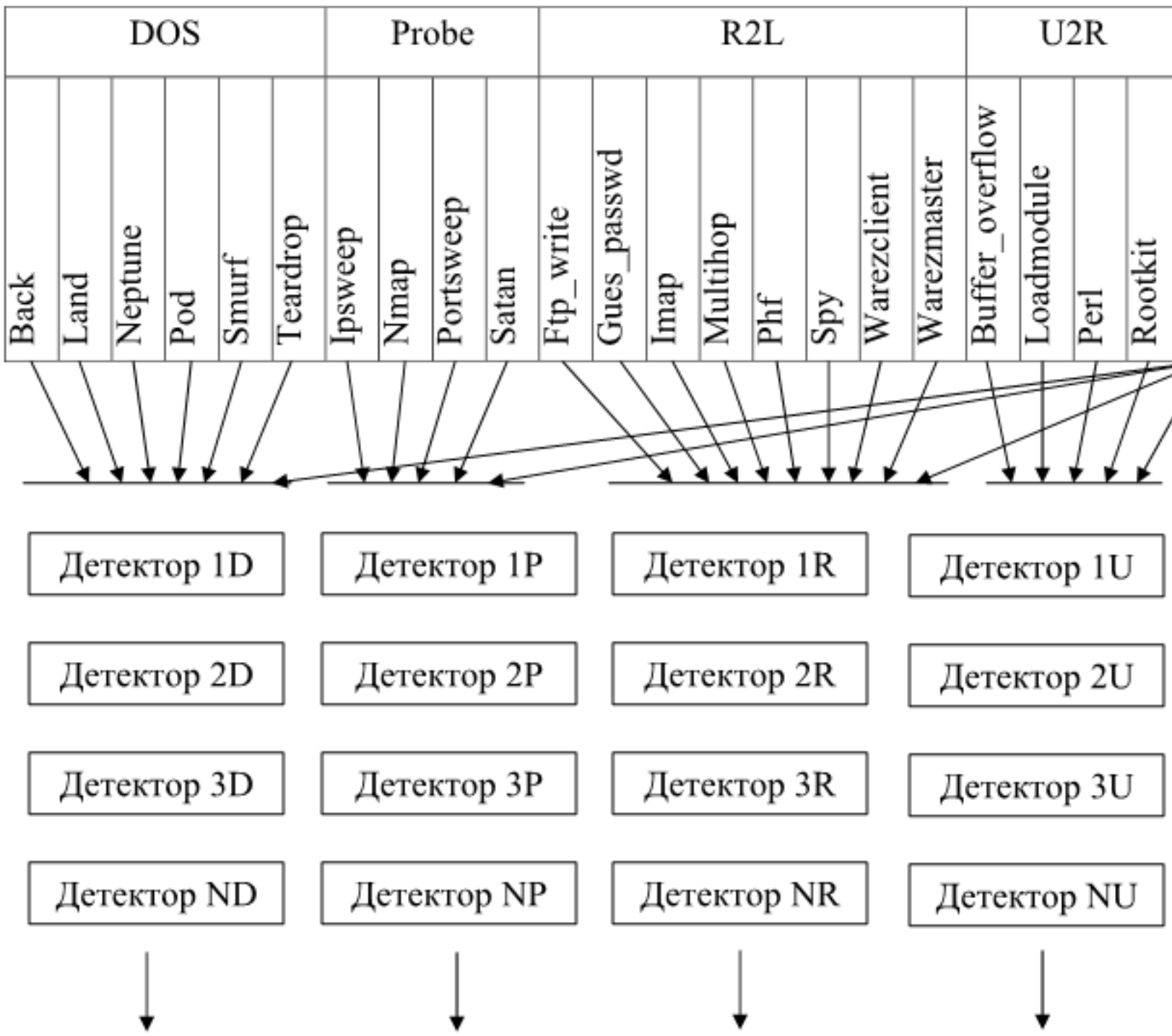


Рисунок 3.2 – Механізм навчання детекторів імунної системи

Як вже наголошувалося, для навчання нейронної мережі і для аналізу мережевого трафіку за допомогою навченої нейронної мережі виділяється 41 параметр з мережевого трафіку [81]. Для навчання нейромережевого детектора (див. розділ 2) використовується навчальна вибірка, яка формується випадковим чином і складається з якогось окремого типу мережевої атаки (наприклад *back* або *land*, що належать до класу *DoS*-атак, або, наприклад, *satan,* який належить до класу *Probe*-атак*)*. Навчається, в загальному випадку, стільки детекторів, скільки існує типів мережевих атак (22 детектори – по одному детектору на кожен з типів мережевих атак).

Слід відзначити, що для виявлення і класифікації атак певного типу може використовуватися не один навчений НІД, а декілька по-різному навчених детекторів, що дозволяє диверсифікувати детектори і відповідно підвищити достовірність виявлення і класифікації атак на ІТМ. В результаті створюється набір різноманітних детекторів для аналізу мережевого трафіку. Проте, перш ніж виконувати аналіз мережевого трафіку, навчені детектори необхідно перевірити на коректність класифікації з метою запобігання виникненню помилкових спрацьовувань. Для цього всі навчені детектори проходять стадію відбору.

3. *Відбір імунних детекторів*. Для мінімізації виникнення помилкових спрацьовувань, коли нормальне з'єднання приймається за атаку, всі навчені НІД проходять перевірку на коректність класифікації. Для цього, на нейронну мережу подається заздалегідь створена тестова вибірка, що складається з параметрів нормального з'єднання. Якщо *i*-й детектор класифікує одне з тестових з'єднань, як атаку, то він знищується, а замість нього генерується і навчається новий детектор. Якщо *i*-й детектор не генерує помилкові спрацьовування на тестовій вибірці, то він вважається коректним і допускається до аналізу вхідного і вихідного мережевого трафіку. В результаті утворюється множина імунних детекторів для аналізу параметрів мережевих з'єднань, яка може поповнюватися, як буде показано далі, за рахунок детекторів імунної пам'яті та генерування нових детекторів після закінчення часу життя.

4. *Функціонування імунних детекторів*. Детектори, які допущені до аналізу мережевого трафіку, утворюють систему виявлення і класифікації мережевих атак. Весь трафік, що отримується комп'ютером, спочатку аналізується сукупністю НІД, і, якщо жоден з детекторів не виявляє аномалію, то трафік обробляється операційною системою і відповідним програмним забезпеченням. Описана система аналізу мережевого трафіку, що складається з сукупності НІД, зображена на рисунку 3.3 [54].

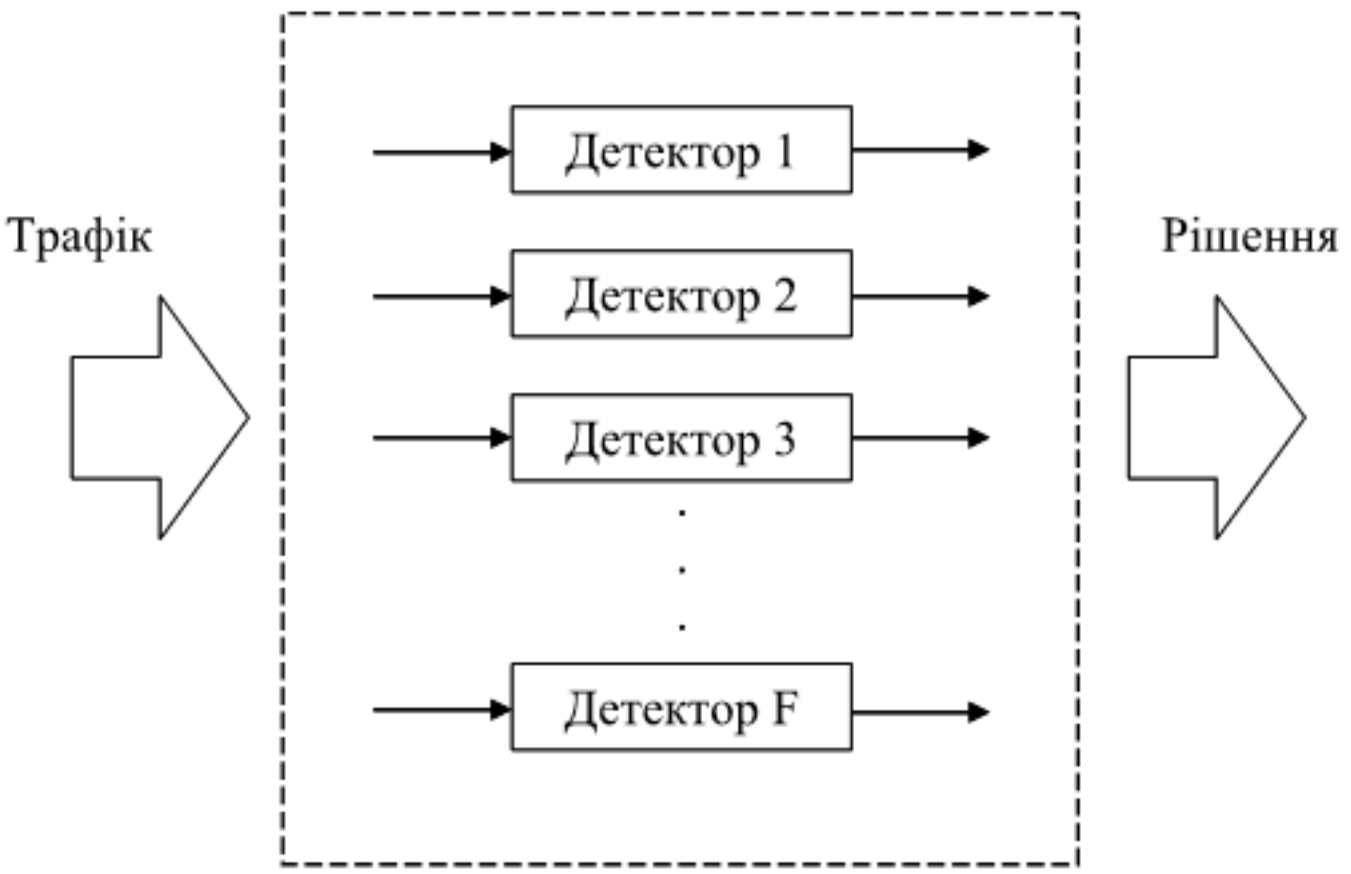


Рисунок 3.3 – Сукупність детекторів для аналізу мережевого трафіку з метою виявлення і класифікації атак

Кожен детектор наділяється часом життя, впродовж якого він аналізує мережевий трафік. Якщо після закінчення виділеного часу детектор не виявив аномалію, він знищується, а на його місце приходить новий детектор, який володіє кращими властивостями виявлення і класифікації атак. Механізм наділення детекторів часом життя дозволяє позбуватися від детекторів, які хоч і пройшли успішно стадії навчання і відбору, проте із-за своєї структурної особливості (набір вагових коефіцієнтів) є малопридатними.

5. *Активація імунних детекторів*. Активація детекторів має на увазі виявлення детектором мережевої атаки. У разі, коли мережеве з'єднання класифікується одним або декількома детекторами як мережева атака, відбувається його блокування, тобто воно не допускається до обробки операційною системою і спеціалізованим програмним забезпеченням. Також видається повідомлення користувачу про спробу атаки комп'ютерної системи.

6. *Формування імунної пам'яті.* При виявленні і блокуванні мережевої атаки доцільним є збереження її параметрів з метою вивчення і детального

аналізу у випадку, якщо така атака не була в базі даних атак. Справа в тому, що НІД навчаються на обмеженому наборі даних, які не можуть містити всі ймовірні мережеві атаки. Для того, щоб підвищити достовірність виявлення і класифікації атак, а також наділити систему виявлення вторгнень гнучкістю і дозволити їй адаптуватися до ситуації, що змінюється, параметри мережевого з'єднання, класифікованого як атака, зберігаються і заносяться в навчальну вибірку, тим самим доповнюючи її актуальними даними. Детектори, які створюватимуться з метою заміни «застарілих» НІД, вже навчатимуться також і на нових даних, що значно дозволить підвищити достовірність виявлення і класифікації атак. Крім цього, на основі детектора, який виявив атаку, створюється новий нейромережевий детектор (операція клонування), який навчається на даних, виділених з виявленої атаки (операція мутації), і потім вводиться в систему виявлення та класифікації атак на ІТМ. Це дозволило точніше виділити дану атаку при повторній подібній атаці на комп'ютерну систему, що захищається, з боку зловмисника. Сукупність детекторів імунної пам'яті зберігає в собі інформацію про всі невідомі мережеві атаки, направлені у минулому на комп'ютерну систему, і забезпечує високий рівень реагування на повторні спроби атак.

Таким чином, введення методики ШІС дозволило розробити гнучку систему, здатну адаптуватися до зміни тенденцій розвитку методів організації мережевих атак, виділити основні характеристики виявлених мережевих атак і навчити на нових даних НІД з метою підвищення достовірності виявлення і класифікації атак на ІТМ. Більш того, детектори імунної пам'яті здатні надійно виявляти повторні спроби атак системи, що захищається, і зберігають інформацію про такі спроби.

Таким чином, в даному підрозділі розроблений комбінований метод виявлення і класифікації атак на ІТМ, який базується на інтеграції нейромережевих детекторів в імунну систему і дозволяє адаптуватися нейромережевій імунній системі до зовнішнього середовища, що змінюється.

3.3 Навчання і функціонування нейромережевих імунних детекторів

Розглянемо навчання НІД і функціонування їх у складі штучної імунної системи [54]. В якості НІД використовується модифікована нейронна мережа векторного квантування, на яку поступає зредукований набір даних відповідно до методу головних компонент (рисунок 3.4).

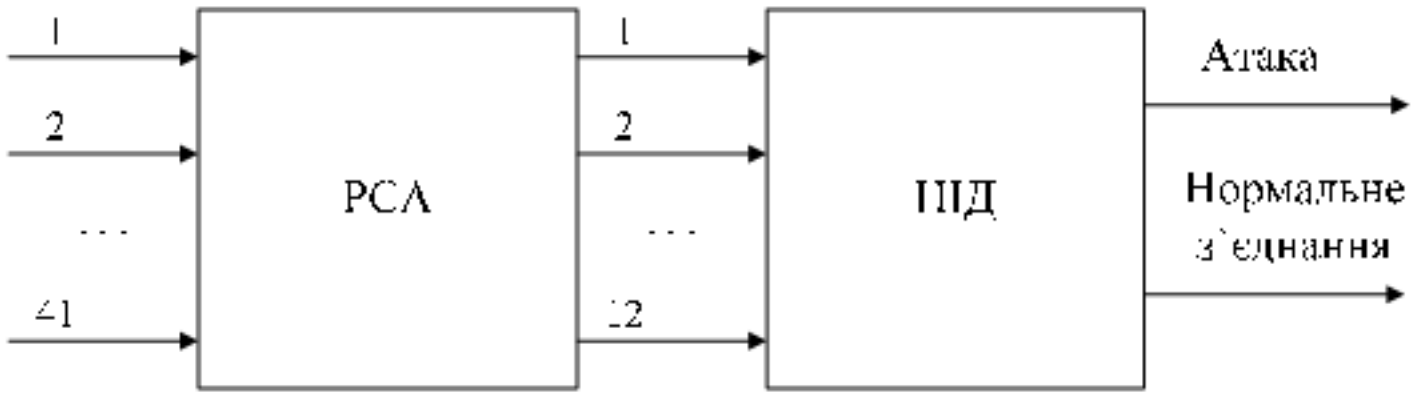


Рисунок 3.4 – Схема навчання і функціонування нейромережевих детекторів у складі штучної імунної системи

Нехай N – множина з’єднань, що відносяться до певного типу мережевих атак, а M – множина з’єднань, що відносяться до класу нормального трафіку. З них випадковим чином формується множина вхідних образів для навчання i-го детектора.

$$X_i = \begin{bmatrix} X_i^1 \\ X_i^2 \\ \dots \\ X_i^L \end{bmatrix} = \begin{bmatrix} X_{i1}^1 & X_{i2}^1 & \dots & X_{in}^1 \\ X_{i1}^2 & X_{i2}^2 & \dots & X_{in}^2 \\ \dots & & & \\ X_{i1}^L & X_{i2}^L & \dots & X_{in}^L \end{bmatrix}. \quad (3.2)$$

Відповідно, множина еталонних образів

$$e_i = \begin{bmatrix} e_i^1 \\ \mathrm{e}_i^2 \\ \dots \\ e_i^L \end{bmatrix} = \begin{bmatrix} e_{i1}^1 & e_{i2}^1 \\ e_{i1}^2 & e_{i2}^2 \\ \dots & \\ e_{i1}^L & e_{i2}^L \end{bmatrix}, \quad (3.3)$$

де L – розмірність навчальної вибірки.

Еталонні вихідні значення для i-го детектора формуються таким чином:

$$e_{i1}^k = \begin{cases} 1, & якщо\ X_i^k \in N \\ 0, & інакше \end{cases}$$
$$e_{i2}^k = \begin{cases} 1, & якщо\ X_i^k \in M \\ 0, & інакше \end{cases} \quad (3.4)$$

Таким чином, для навчання імунних детекторів виявлення одного і того ж типу атаки використовуються різні навчальні вибірки, які генеруються випадковим чином з множини з'єднань, що складаються з певного типу мережевих атак і нормального трафіку. В результаті створюється множина різноманітних детекторів, які здатні виявляти і класифікувати різні типи атак.

У загальному випадку, методику створення і навчання НІД для виявлення і класифікації певного типу атаки можна представити в наступному вигляді:

1. Створюємо нейромережевий детектор з випадковою ініціалізацією вагових коефіцієнтів.

2. Випадковим чином з набору параметрів мережевих з'єднань вибираємо N з'єднань, що відносяться до певного типу мережевих атак, і M з'єднань, що відносяться до класу нормального трафіку, кількість яких становить відповідно 80% і 20%.

3. Послідовно подаємо вибрані параметри з'єднань на нейронну мережу і здійснюємо модифікацію вагових коефіцієнтів. Вагові коефіцієнти коректуються згідно наступного:

a) обчислюється евклідова відстань між вхідним образом і ваговими векторами нейронних елементів шару Кохонена за формулою (2.6);

b) визначається нейронний елемент-переможець з номером k за формулою (2.12);

c) проводиться модифікація вагових коефіцієнтів нейрона-переможця

відповідно до формули (2.13), якщо при подачі на вхід мережі параметрів мережевої атаки переможцем є один з перших *f* нейронів нейронної мережі або при подачі на вхід мережі параметрів нормального з'єднання переможцем є один з *l* останніх нейронів мережі Кохонена. Інакше, згідно формули (2.14).

4. Процес повторюється, починаючи з пункту 3 для всіх вхідних образів. Навчання нейронної мережі проводиться до бажаного ступеня узгодження між вхідними і ваговими векторами, тобто до тих пір, поки значення сумарної квадратичної помилки не стане мінімальним:

$$E_i = \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{2}(Y_j^k - e_j^k)^2, \tag{3.5}$$

де Y_j^k – *j*-е вихідне значення нейромережевого детектора для *k*-го вхідного образу;

e_j^k – *j*-е еталонне значення для *k*-го еталонного образу.

5. Процес повторюється, починаючи з пункту 1 до тих пір, поки кількість навчених імунних детекторів не стане рівною заданому значенню *F*.

Загальний алгоритм методу виявлення і класифікації атак на ІТМ на базі штучних імунних систем і нейронних мереж можна представити в наступному вигляді [54]:

1. Створення імунного детектора на базі багатошарової нейронної мережі з одним прихованим шаром Кохонена і початкова ініціалізація вагових коефіцієнтів.

2. Формування навчальної вибірки для створеного детектора. Навчальна вибірка формується шляхом вибору випадковим чином *N* з'єднань, що відносяться до певного класу мережевої атаки, і *M* з'єднань, що відносяться до нормального трафіку.

3. Послідовна подача даних з навчальної вибірки на нейромережевий детектор і навчання його згідно правилу «переможець бере все» і формул

(2.6), (2.12)–(2.14).

4. Навчений детектор перевіряється на тестовій вибірці, що складається з параметрів нормальних з'єднань. Якщо детектор коректно класифікує подані дані, то він «допускається» до аналізу мережевого трафіку. Якщо ж детектор класифікує тестові дані, що подаються, як мережеву атаку, то він знищується.

5. Впровадження навченого і відібраного детектора в підсистему аналізу мережевого трафіку.

6. Якщо відведений час життя детектора закінчився і детектор не виявив аномалію в мережевому трафіку, то він знищується, і процес починається з пункту 1.

7. Якщо детектор виявив аномалію, то відбувається його активація.

8. Мережеве з'єднання, що класифікується як атака, блокується, а користувачеві видається повідомлення.

9. Виділення і аналіз параметрів мережевого з'єднання, класифікованого як атака, і занесення виділених параметрів в базу для навчання нових імунних детекторів.

10. Генерація детектора імунної пам'яті і навчання його на параметрах виявленої мережевої атаки з метою підвищення достовірності виявлення і класифікації можливих аналогічних атак на систему, що захищається, і впровадження даного детектора в підсистему аналізу мережевого трафіку.

3.4 Експериментальні дослідження комбінованого методу виявлення і класифікації атак

Проведемо експериментальні дослідження комбінованого методу виявлення і класифікації атак на ІТМ на основі методів штучних імунних систем і нейронних мереж і порівняємо їх з результатами, представленими в другому розділі. Експериментальні дослідження проводилися на основі методики представленої в таблиці 2.7.

Тестування НІД проводилося за наступною схемою [54]:

– спочатку здійснюється створення імунного детектора, в основі якого лежить нейронна мережа з одним прихованим шаром Кохонена (див. рисунок 2.5). Кількість нейронів вхідного шару дорівнює кількості головних компонент параметрів з'єднання (згідно аналізу, проведеного в розділі 2, оптимальною кількістю аналізованих головних компонент є 12). Кількість нейронів в прихованому шарі дорівнює 10, причому перші 8 нейронів характеризують мережеву атаку, а останні два характеризують нормальне з'єднання. Кількість вихідних нейронів дорівнює два. Активність першого вихідного нейрона класифікує мережеве з'єднання як атаку, а активність другого – нормальне з'єднання. Описана нейронна мережа створюється з випадковою ініціалізацією вагових коефіцієнтів;

– проводиться навчання створеного детектора на підготовленій спеціально для нього навчальній вибірці. Навчальна вибірка формується за допомогою випадкового вибору параметрів з'єднання певної мережевої атаки і параметрів нормального з'єднання з бази KDD Cup 1999 Data [81]. Навчання проводиться до досягнення бажаного ступеня відповідності між вхідними і вихідними даними нейронної мережі;

– перевірка коректності функціонування і відбір імунних детекторів, що пройшли навчання. Коректність класифікації образів проводиться на спеціально сформованій тестовій вибірці, яка складається з параметрів нормального мережевого трафіку. Якщо детектор, що перевіряється, класифікує тестовий трафік як атаку, він знищується і генерується новий детектор, який також навчається і проходить відбір. Описаний процес повторюється до тих пір, поки не буде сформований коректний детектор;

– детектор, який пройшов перевірку, аналізує дані мережевого трафіку.

Розглянемо механізм відбору нейромережевих детекторів, що пройшли стадію навчання, який дозволяє позбутися від виникнення помилок другого роду, тобто тих помилок, коли нормальне з'єднання класифікується як

мережева атака.

У таблицях 3.1 і 3.2 представлено 20 НІД, які пройшли стадію навчання і знаходяться на стадії відбору. У таблиці 3.1 представлені детектори, які навчалися на атаках класу *dos_neptune*, а в таблиці 3.2 – детектори, які навчалися на атаках класу *probe_portsweep*.

Проаналізуємо отримані результати. Як видно з таблиць, деякі детектори (*D1-D10, P1, P5,P7*) достатньо точно виявляють мережеві атаки, на яких відбувалося їх навчання. Проте, при розгляді показника FPR, який відображає рівень помилок другого роду, видно, що він для всіх представлених детекторів в таблицях 3.1–3.2 дуже великий, і такі детектори генеруватимуть помилкові спрацьовування досить часто, що є недопустимим для системи виявлення вторгнень. Такі детектори повинні знищуватися на стадії відбору, результатом функціонування якої повинні бути детектори з низьким рівнем помилок другого роду.

Таблиця 3.1 – Відбір НІД D_i

№	TPR (Se),%	TNR (Sp),%	FNR,%	FPR,%	Accu,%
Детектор D1	100	94,5	0	5,5	97,3
Детектор D2	100	95,6	0	4,4	97,8
Детектор D3	100	95,0	0	5,0	97,5
Детектор D4	100	95,8	0	4,2	97,9
Детектор D5	100	94,7	0	5,3	97,4
Детектор D6	100	99,0	0	1,0	99,5
Детектор D7	100	95,1	0	4,9	97,6
Детектор D8	100	98,7	0	1,3	99,4
Детектор D9	100	98,3	0	1,7	99,2
Детектор D10	100	94,4	0	5,6	97,2

Слід зазначити узагальнюючу властивість НІД. Так, навчений на даних певного класу атак детектор здатний виявляти також атаки, що належать іншому класу. Це відбувається через те, що нормальне мережеве з’єднання за своїми параметрами часто відрізняється від мережевої атаки і вибрана нейронна мережа дозволяє відстежувати ці відмінності.

Таблиця 3.2 – Відбір НІД P_i

№	TPR (Se),%	TNR (Sp),%	FNR,%	FPR,%	Accu,%
Детектор P1	100	92,2	0	7,8	96,1
Детектор P2	99,6	93,9	0,4	6,1	96,8
Детектор P3	99,8	92,8	0,2	7,2	96,3
Детектор P4	99,3	98,1	0,7	1,9	98,7
Детектор P5	99,9	95,8	0,1	4,2	97,9
Детектор P6	99,4	94,0	0,6	6,0	96,7
Детектор P7	99,9	93,0	0,1	7,0	96,5
Детектор P8	99,0	94,8	1,0	5,2	96,9
Детектор P9	99,1	94,9	0,9	5,1	97,0
Детектор P10	98,7	95,5	1,3	4,5	97,1

Результати проведених експериментів по дослідженню узагальнюючої властивості НІД, що дозволяє одному детектору виявляти різні типи атак, представлені в таблицях 3.3–3.4.

Таблиця 3.3 – Результати виявлення мережевих атак детекторами 1–3

	Детектор 1 (навчений на *DoS_Back*)	Детектор 2 (навчений на *DoS_Neptune*)	Детектор 3 (навчений на *DoS_Teardrop*)
Тип атаки	Достовірність виявлення		
DoS-атаки			
Back	100,0	0,0	0,0
Land	0,0	100	81,0
Neptune	99,1	100	100
Pod	0,0	0,0	31,1
Smurf	0,0	0,0	0,2
Teardrop	0,0	2,7	100
Probe-атаки			
Ipsweep	0,1	6,6	7,0
Nmap	80,5	0,0	96,1
Portsweep	2,1	98,9	97,8
Satan	13,3	88,8	93,9
R2L-атаки			
Ftp_write	0,0	0,0	0,0
Guess_passwd	0,0	94,4	0,0
Imap	0,0	83,4	16,7
Multihop	0,0	0,0	0,0
Phf	0,0	0,0	0,0

Продовження таблиці 3.3

Spy	100,0	0,0	50,0
Warezclient	1,1	0,3	0,0
Warezmaster	0,0	0,0	0,0
U2R-атаки			
Buffer_overflow	0,0	0,0	0,0
Loadmodule	88,9	0,0	11,2
Perl	33,4	0,0	0,0
Rootkit	0,0	0,0	30,0

Таблиця 3.4 – Результати виявлення мережевих атак детекторами 4–6

	Детектор 4 (навчений на *Probe_Nmap*)	Детектор 5 (навчений на *Probe_Portsweep*)	Детектор 6 (навчений на *R2L_Ftpwrite*)
Тип атаки	Достовірність виявлення		
DoS-атаки			
Back	99,1	0,0	0,3
Land	9,5	100	23,8
Neptune	99,9	100	0,0
Pod	12,9	0,0	1,9
Smurf	0,1	0,0	0,0
Teardrop	11,0	8,8	0,0
Probe-атаки			
Ipsweep	5,7	7,0	1,0
Nmap	100	0,0	0,0
Portsweep	30,8	100	0,1
Satan	96,1	92,2	2,1
R2L-атаки			
Ftp_write	0,0	0,0	100
Guess_passwd	0,0	1,9	5,7
Imap	0,0	75,0	0,0
Multihop	0,0	0,0	57,2
Phf	0,0	0,0	0,0
Spy	100	0,0	0,0
Warezclient	0,6	0,2	65,0
Warezmaster	0,0	0,0	90,0
U2R-атаки			
Buffer_overflow	0,0	3,4	83,4
Loadmodule	100	0,0	0,0
Perl	0,0	0,0	0,0
Rootkit	0,0	0,0	20,0

Кожен з представлених детекторів навчався на одному певному типі

атак (так, детектор 2 навчався на атаці типу *DoS_Neptune*). Після навчання детектори класифікували всі вхідні образи з тестової вибірки. Як видно з представлених таблиць, крім того, що навчені детектори з ймовірністю 100% виявляють той тип атак, на яких відбувалося навчання, вони ще і виявляють атаки з абсолютно інших класів. Причому, деякі типи атак вони виявляють з високою ймовірністю. Звідси випливає, що після навчання і відбору НІД характеризуються властивістю виявляти не тільки відомі, але і невідомі мережеві атаки.

Таким чином, відбувається перехресна перевірка вхідного трафіку всіма детекторами, кожен з яких робить внесок до прийняття рішення про небезпеку або безпеку даного з'єднання, що підвищує надійність системи. Окрім цього, як показали результати експериментів, НІД здатні виявляти не тільки вже відомі атаки, але і невідомі, що є важливою вимогою до сучасних систем безпеки.

Як показано вище, один детектор виявляє різні типи атак, причому навіть ті, які не входили до складу навчальної вибірки. Для оцінки виявляючої властивості детекторів проведемо аналіз залежності виявлення кількості мережевих атак від кількості активних НІД.

У таблиці 3.5 представлені результати виявлення мережевих атак десятьма різними детекторами.

Таблиця 3.5 – Кількість виявлених атак

Кількість детекторів	Кількість виявлених атак
1	108810
2	109060
3	112214
4	162898
5	163235
6	164087
7	164131
8	164143
9	164143
10	164849

Таблиця організована таким чином, що відображає кількість виявлених атак залежно від кількості детекторів. Так, наприклад, детектор під номером 1 виявляє *108810* мережевих атак, два детектори разом вже виявляють *109060* різних атак, три детектори – *112214* і т. д.

На рисунку 3.5 приведена експериментальна ймовірність виявлення мережевої атаки залежно від кількості детекторів.

Як видно з графіка, на перших етапах додавання нового детектора суттєво впливає на достовірність виявлення мережевих атак (стрибкоподібна поведінка на графіку). Це відбувається через те, що кожен окремий детектор виявляє різні класи атак. Проте, далі із збільшенням кількості детекторів, кількість мережевих атак, що виявляються, росте вже не стрибкоподібно, а плавно, поступово до 100%. Це відбувається через те, що нові детектори, в основному, виявляють ті мережеві атаки, які вже були виявлені іншими детекторами, і тільки завдяки тому, що різні детектори навчалися на різних, відмінних від інших даних, відбувається виявлення «специфічних» атак і плавне зростання загальної кількості виявлених атак.

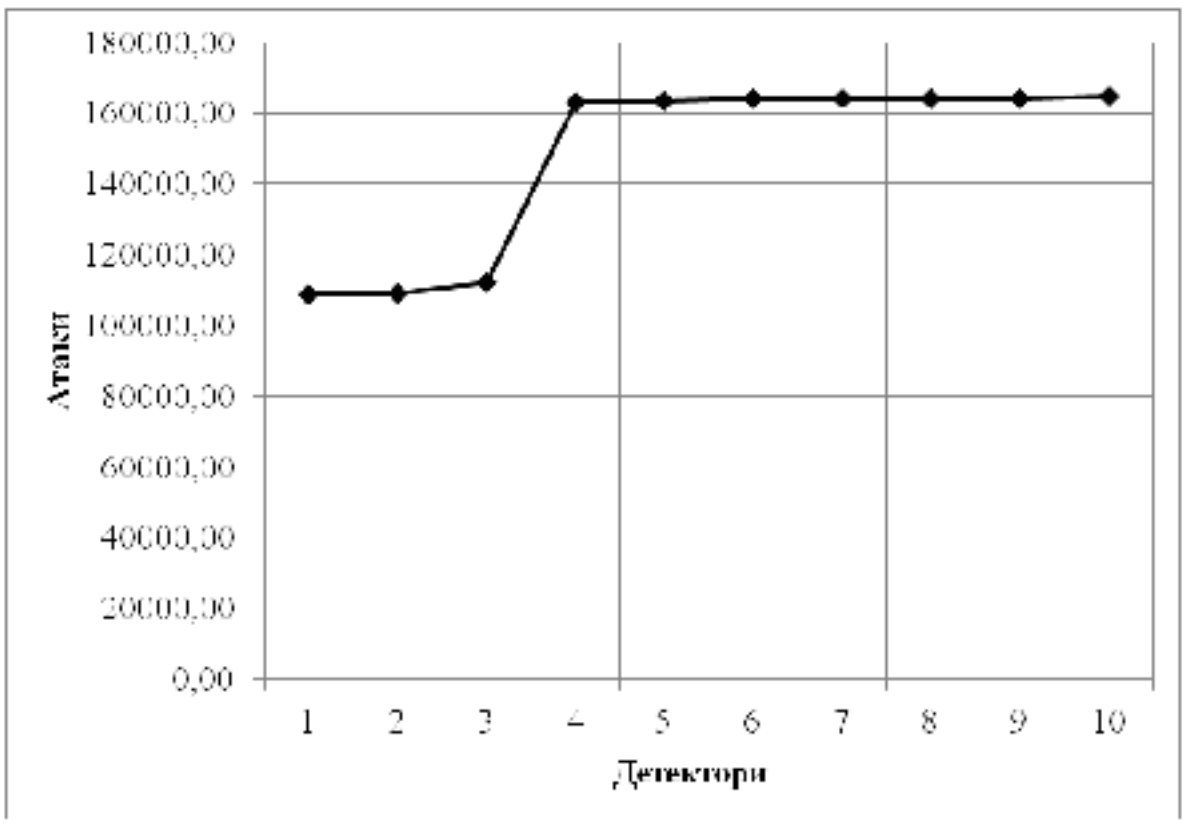


Рисунок 3.5 – Залежність кількості виявлених атак від кількості детекторів

Проведемо експериментальні дослідження механізму адаптації НІД.

У таблиці 3.6 представлені результати адаптації НІД до нових атак в результаті операцій клонування і мутації. В якості батьківського взято детектор, який навчався на атаці типу *dos_land*. Даний детектор 2 виявив невідомі для нього атаки – *r2l_imap* і *probe_portsweep*. Припустимо, що таких атак немає в базі даних, і згенеруємо два нових детектори *A1* і *A2*. Додамо параметри виявлених атак в навчальні вибірки для цих детекторів і навчимо відповідні детектори-клони. Таблиця 3.6 містить характеристики даних детекторів.

Таблиця 3.6 – Адаптація НІД *Ai*

	Детектор 2, Sp(TNR)= 99,0%	Детектор A1, Sp(TNR)= 99,1%	Детектор A2, Sp(TNR)= 98,9%
Тип атаки	*Se (TPR),%*	*Se (TPR),%*	*Se (TPR),%*
DoS-атаки			
Back	0,0	0,0	0,0
Land	100,0	100,0	100,0
Neptune	80,9	80,1	80,9
Pod	2,3	0,0	31,8
Smurf	0,0	0,0	0,0
Teardrop	0,0	2,7	0,0
Probe-атаки			
Ipsweep	7,22	0,2	33,9
Nmap	0,0	0,0	0,0
Portsweep	15,9	2,6	55,3
Satan	11,0	31,3	11,0
R2L-атаки			
ftp_write	0,0	0,0	0,0
guess_passwd	3,8	1,9	5,7
Imap	83,3	91,7	83,3
Multihop	0,0	0,0	14,3
Phf	0,0	0,0	0,0
Spy	0,0	0,0	0,0
Warezclient	0,2	1,8	0,2
Warezmaster	0,0	10,0	0,0
U2R-атаки			
buffer_overflow	0,0	0,0	0,0
Loadmodule	0,0	0,0	0,0
Perl	0,0	66,7	0,0
Rootkit	0,0	20,0	0,0

Аналізуючи результати, представлені в таблиці 3.6, приходимо до висновку, що НІД здатні до адаптації і покращення властивостей виявлення мережевих атак. Так, детектор А1 в 2,8 раза почав краще виявляти атаки класу *probe_satan*, на 8,4% краще атаку типу *r2l_imap*, а також почав виявляти атаки класів *r2l_warezmaster, u2r_perl* і *r2l_rootkit*. А детектор А2 показав кращі результати на атаках класів *dos_pod, probe_ipsweep, probe_portsweep, r2l_multihop*.

Таким чином, експериментально підтверджено, що НІД здатні до адаптації до нових атак на ІТМ. Інтеграція нейромережевих детекторів в ШІС дозволила добитися підвищення достовірності виявлення і класифікації атак на ІТМ, а також дала можливість зробити систему більш гнучкою і здатною до самонавчання. Це дозволяє системі розвиватися і виявляти нові мережеві атаки на інформаційні ресурси.

Висновки до розділу 3

1. Побудова комбінованого методу виявлення і класифікації атак на ІТМ відбувалася на основі базових принципів і механізмів біологічної імунної системи: генерація і навчання імунних детекторів, відбір детекторів, функціонування детекторів, активація і адаптація детекторів до невідомих атак, формування імунної пам’яті.

2. Розроблено комбінований метод виявлення і класифікації атак на ІТМ, який базується на інтеграції нейромережевих детекторів в ШІС, що дає можливість НІД еволюціонувати з метою ефективного виявлення та класифікації атак на комп’ютерні системи. Розроблено методику функціонування НІД, яка дозволяє з високою точністю виявляти мережеві атаки, а також зберігати інформацію в детекторах імунної пам’яті про минулі атаки на комп’ютерну систему. Також дозволяє виявляти нові мережеві атаки, вивчати їх та виділяти сигнатури з метою навчання нових детекторів для виявлення і класифікації атак.

3. Проведені експериментальні дослідження підтверджують ефективність запропонованого підходу, оскільки НІД можуть адаптуватися до нових атак на ІТМ за рахунок здійснення операцій клонування і мутації з метою підвищення достовірності їх виявлення і класифікації.

РОЗДІЛ 4
ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВИЯВЛЕННЯ І КЛАСИФІКАЦІЇ АТАК НА ІНФОРМАЦІЙНІ ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ

4.1 Структура програмного забезпечення та алгоритми функціонування нейромережевої імунної системи виявлення і класифікації атак

Для ефективної організації процесу виявлення і класифікації атак на ІТМ на основі модульного принципу програмування розроблені програмні засоби, що дозволяють здійснити основні операції виявлення і класифікації атак [53, 56].

На рисунку 4.1 представлена схема програмного забезпечення нейромережевої імунної системи виявлення і класифікації атак на ІТМ.

Рисунок 4.1 – Схема програмного забезпечення нейромережевої імунної системи виявлення і класифікації атак на ІТМ

Запропонована схема складається з наступних основних модулів:

1. Модуль обробки інформації при виявленні і класифікації атак.

2. Модуль генерації НІД.
3. Модуль навчання НІД.
4. Модуль відбору НІД.
5. Модуль виявлення і класифікації атак на ІТМ.
6. Модуль адаптації НІД.

Розглянемо детальніше кожен з описаних модулів.

Модуль обробки інформації призначений для представлення параметрів мережевих з'єднань у зручному для аналізу вигляді. Він складається з двох модулів – модуля захоплення мережевого трафіку в комп'ютерній мережі і модуля обчислення головних компонент (PCA) з параметрів захопленого трафіку [56].

Для захоплення мережевого трафіку використовується спеціалізоване програмне забезпечення, так званий сніффер (sniffer) [185]. Захоплений сніффером мережевий трафік аналізується – з нього виділяється 41 параметр мережевого з'єднання, які характеризують дане з'єднання і містять час роботи з'єднання, тип протоколу, тип служби, кількість переданих байт і т. д.

Оскільки різні параметри мережевого з'єднання мають різний тип інформації (наприклад, параметр «час роботи з'єднання» задається в секундах, «тип протоколу» - у символьному вигляді, а «кількість байт від джерела до приймача» задається в байтах), то для того, щоб аналізувати такі різнорідні дані, їх треба привести до загального вигляду. Перш ніж поступити на вхід модуля обчислення головних компонент параметри мережевого з'єднання піддаються процедурі перетворення згідно таблиці 2.1.

Як було показано в другому розділі, 41 параметр мережевого трафіку є зайвими, і для успішного аналізу трафіку з метою виявлення і класифікації мережевих атак необхідно зменшити кількість аналізованих параметрів. Для зменшення розмірності аналізованої інформації використовується модуль, що здійснює виділення головних компонент.

У таблиці 4.1 представлені значення 41 параметра п'яти мережевих з'єднань.

Таблиця 4.1 – Значення параметрів мережевого трафіку до обчислення головних компонент

З'єднання 1	З'єднання 2	З'єднання 3	З'єднання 4	З'єднання 5
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
181	256	370	232	302
5450	1273	520	8766	2164
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
8	17	3	10	16
8	17	3	10	16
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
100	100	100	100	100
0	0	0	0	0
0	0	0	0	0
9	74	82	31	231
9	209	255	255	255
100	100	100	100	100
0	0	0	0	0
11	1	1	3	0
0	3	4	4	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Модуль PCA є програмною реалізацією математичного методу обчислення головних компонент і призначений для зменшення розміру аналізованої інформації, що приводить до підвищення достовірності виявлення і класифікації мережевих атак і швидкості аналізу мережевих пакетів. На вхід модуля поступають дані із сніффера – 41 параметр мережевого з'єднання. На виході модуля PCA, після виконання всіх перетворень і обчислень, отримуємо дані, які є 12 головними компонентами. Як було показано в другому розділі, у 12 головних компонентах параметрів мережевого трафіку міститься 99,9% інформації [56].

У таблиці 4.2 представлені дані по 41 параметрі п'яти мережевих з'єднань, 12 з яких (виділені кольором) містять повну інформацію про дане з'єднання і використовуються для аналізу НІД.

В результаті, на вхід модуля обробки інформації подається мережеве з'єднання у дійсній формі, а на виході з модуля отримуємо дані, які знаходяться в зручній для аналізу формі та передаються в модуль виявлення і класифікації атак.

Модулі створення, навчання і відбору детекторів призначені для створення НІД, які є основними елементами системи виявлення і класифікації атак [56]. Кожен окремий імунний детектор є штучною нейронною мережею (див. розділ 2). Такий НІД складається з трьох шарів нейронних елементів. Перший шар нейронних елементів є розподільним. Функцією його є розподіл параметрів мережевого з'єднання на нейронні елементи другого шару. Другий шар складається з нейронів Кохонена і здійснює кластеризацію вхідного простору образів. Третій шар складається з двох лінійних нейронних елементів, які здійснюють відображення кластерів, сформованих шаром Кохонена, в два класи, які характеризують клас нормальних з'єднань і клас мережевих атак.

Опишемо роботу даного модуля, починаючи із створення НІД і закінчуючи впровадженням коректного детектора в підсистему виявлення і класифікації атак на ІТМ.

Таблиця 4.2 – Значення параметрів мережевого трафіку після обчислення головних компонент

З'єднання 1	З'єднання 2	З'єднання 3	З'єднання 4	З'єднання 5
0,763701	0,981259	1,03307	1,059754	0,974645
0,669182	0,537566	0,527792	0,546017	0,406942
-0,73592	-0,81945	-0,85446	-0,88031	-0,73416
-0,34766	0,08751	0,166642	0,16134	0,19282
0,437885	0,132695	0,093223	0,26349	-0,39345
-0,33832	-0,14563	-0,0895	-0,14361	0,012605
-0,16508	-0,15383	-0,16389	-0,18696	-0,08801
0,212952	0,103092	0,070628	0,139702	-0,1396
-0,32469	-0,02379	0,053603	0,076463	-0,02222
-0,32539	-0,09706	-0,02025	-0,02862	-0,00613
0,360893	0,030881	-0,03733	-0,06778	0,043152
-0,16864	-0,03566	-0,00599	-0,00011	-0,02076
-0,08027	-0,01062	0,010248	0,01271	-0,00111
-0,0089	-0,00332	-0,00089	-0,00453	0,008783
-0,04007	-0,01542	-0,00583	-0,01124	0,002863
0,129852	0,033054	0,004976	0,011304	-0,01009
-0,06122	-0,00133	0,011071	-0,00143	0,017338
-0,0056	-0,00188	-0,00126	-0,00405	0,004164
0,032095	0,016327	0,009496	0,007509	0,008185
0,001432	-0,00089	0,005549	0,006202	-0,00239
-0,00145	-0,00127	-0,00129	-0,0023	0,001286
-0,0012	-0,00089	-0,00064	-0,00094	-0,00014
-0,0061	-0,00027	0,001456	0,001467	0,001352
0,00857	9,38E-05	-0,00152	-0,00174	-0,00034
-0,00567	-0,00044	0,000522	0,000799	-0,00027
-0,00996	-0,00352	-0,00142	-0,00175	-0,0006
0,000258	0,000279	0,000232	0,000639	-0,00089
-0,00082	-0,00014	0,00014	-0,00014	0,00087
-0,00255	-0,00088	-0,00041	0,001028	-0,00026
-0,00034	0,000499	0,000743	0,000345	0,000148
0,003002	0,000834	0,000268	0,000303	-0,00015
0,000322	2,03E-05	-2,66E-06	-4,70E-05	-5,59E-05
-0,00065	-0,00054	-0,00049	-0,00046	-0,00079
-0,00105	-0,00047	-0,00032	-0,00014	-0,00063
0,000703	0,000179	2,34E-05	0,000152	-0,00024
-0,00194	-0,00046	-8,78E-05	-0,0001	-0,00016
8,09E-05	-8,26E-06	-2,71E-05	-5,61E-05	6,40E-05
-9,04E-05	-1,81E-05	-8,92E-07	-1,65E-05	0,000103
0,000148	1,45E-05	-1,74E-05	-2,28E-05	9,88E-07
1,05E-19	2,39E-20	4,40E-21	5,37E-21	2,46E-21
-7,02E-20	-1,58E-20	-2,69E-21	-3,40E-21	-1,32E-21

Модуль створення детекторів призначений для генерації нейронних мереж, які і складають основу детекторів. На вхід даного модуля подаються

такі дані, як: кількість нейронів в першому шарі і кількість нейронів у прихованому шарі. Далі модуль генерує нейронну мережу із заданими параметрами та ініціалізує вагові коефіцієнти між нейронними елементами згідно випадкового розподілу.

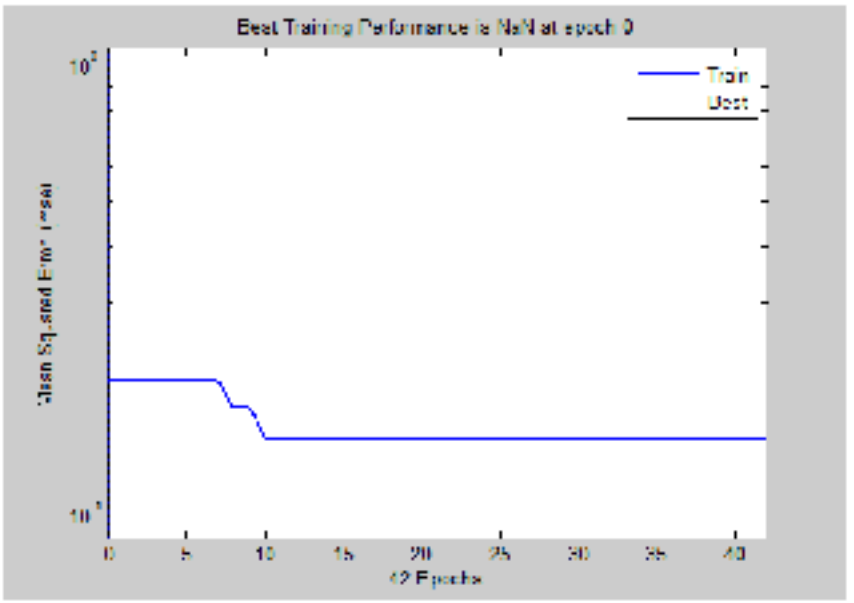
Згенерована нейронна мережа подається на вхід модуля навчання детекторів, завданням якого є навчити детектор коректно виявляти і класифікувати образи нормальних мережевих з'єднань і мережевих атак. Для цього на вхід модуля навчання також подаються дані з навчальної вибірки, які і складають навчальну вибірку для навченого детектора. Слід зазначити, що дані з навчальної вибірки для навчання кожної нейронної мережі вибираються випадковим чином і, отже, є унікальними для кожної окремої нейронної мережі, що забезпечує велику структурну різноманітність НІД.

Для навчання однієї нейронної мережі використовуються дані, що складаються з 64 з'єднань одного з чотирьох класів мережевих атак (що складає 80 відсотків зі всіх навчальних даних для нейронної мережі) і 16 з'єднань, що відносяться до класу нормальних мережевих з'єднань (що складає 20 відсотків навчальної вибірки). Дане співвідношення класів в навчальній вибірці було отримане експериментальним шляхом і показало якнайкращі результати.

В результаті, кожна нейронна мережа навчається на параметрах 80 мережевих з'єднань. При навчанні нейронної мережі використовується контрольоване конкурентне навчання відповідно до правила «переможець бере все». Тобто дані з навчальної вибірки, сформованої спеціально для даної нейронної мережі, послідовно подаються на її вхід і залежно від узгодженості поданих даних і даних на виході нейронної мережі коректуються вагові коефіцієнти згідно формул (2.13) і (2.14).

Процес навчання нейронної мережі проводиться до бажаного ступеня узгодження між вхідними і ваговими векторами. Якщо в процесі навчання не відбувається досягнення бажаного значення середньоквадратичної помилки (див. формула 3.5), то вважається, що нейронна мережа не здатна навчитися,

і вона знищується. На рисунку 4.2 відображені графіки зміни середньоквадратичної помилки в процесі навчання нейронної мережі.



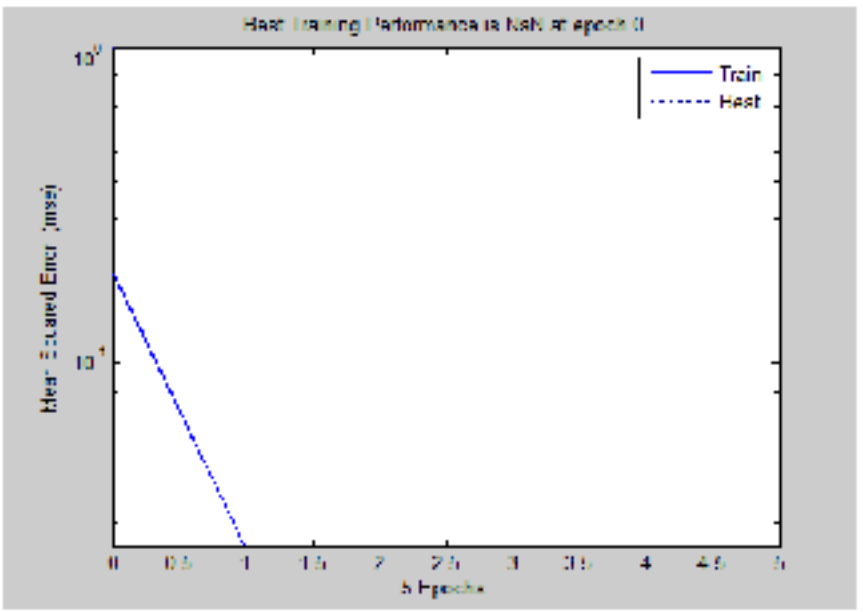


Рисунок 4.2 – Графіки зміни середньоквадратичної помилки в процесі навчання нейронної мережі

Час навчання однієї нейронної мережі складає в середньому 1–3 секунди, що є хорошим показником.

Навчені нейронні мережі повинні пройти процес верифікації з метою підтвердження їх коректності при класифікації різних образів мережевого трафіку. Функцію перевірки коректності функціонування навчених нейронних мереж виконує модуль відбору детекторів.

Навчена нейронна мережа перевіряється на спеціально підготовленій тестовій вибірці, що складається з параметрів нормальних мережевих з'єднань. Нейронна мережа аналізує і класифікує дані з тестової вибірки і,

якщо вона виявляє в ній мережеву атаку, то вважається некоректною (оскільки тестова вибірка містить тільки параметри нормальних з'єднань) і знищується. Якщо нейронна мережа не виявляє на тестовій вибірці мережевих атак, то вона «проходить» стадію відбору, стає НІД і впроваджується в підсистему виявлення і класифікації атак.

Механізм верифікації функціонування нейронних мереж, що пройшли стадію навчання, дозволяє позбутися від виникнення помилок другого роду, тобто тих помилок, коли нормальне з'єднання класифікується як мережева атака.

Схема алгоритму роботи модулів створення, навчання і відбору НІД представлена на рисунку 4.3.

Рисунок 4.3 – Схема алгоритму роботи модулів створення, навчання і відбору детекторів

НІД, які успішно пройшли стадії навчання і відбору, складають основу модуля виявлення і класифікації мережевих атак. Сукупність НІД, які аналізують мережевий трафік, представлена на рисунку 4.4 [56].

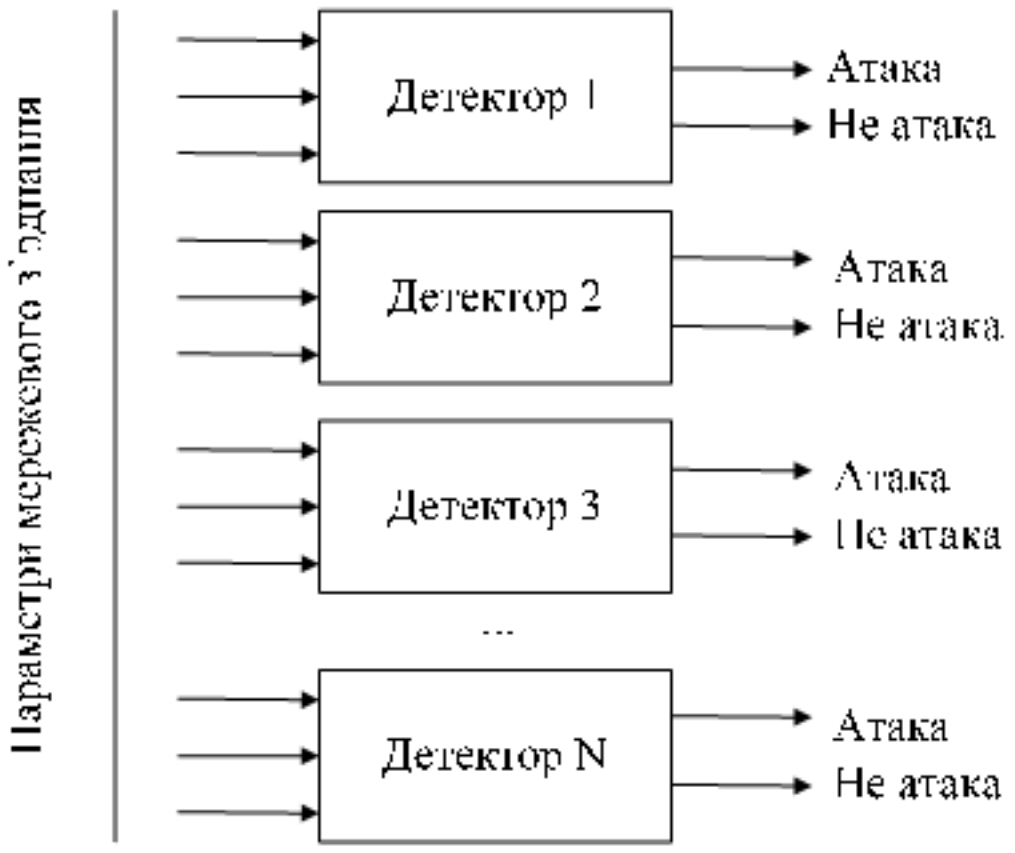


Рисунок 4.4 – Сукупність НІД для виявлення і класифікації атак

Як було зазначено, при навчанні НІД використовується навчальна вибірка, що складається з параметрів якогось окремого типу мережевих атак (наприклад, *DoS*-атак*)* і нормального з'єднання. Таким чином, формується окремий детектор, який налаштований виявляти і класифікувати мережеві атаки певного типу. Сукупність же НІД забезпечує виявлення і класифікацію мережевих атак, що належать до будь-якого з класів і типів.

Для виявлення і класифікації атак на ІТМ паралельно на вхід кожного з функціонуючих НІД з модуля обробки інформації подаються 12 параметрів мережевого з'єднання. Детектори аналізують ці параметри і приймають відповідне рішення. Якщо всі детектори визначили аналізоване з'єднання як нормальне (не атака), то воно допускається до обробки і виконання відповідним програмним забезпеченням (ПЗ) і операційною системою (ОС). Якщо ж хоч би один з детекторів класифікував поточне з'єднання як мережеву атаку, то воно блокується і видається повідомлення про те, що на комп'ютерну систему, яка захищається, проводиться мережева атака, і що

небезпечне мережеве з’єднання буде завершено.

Таким чином, функціонування модуля виявлення і класифікації мережевих атак можна представити у вигляді наступного алгоритму (рисунок 4.5) [56]:

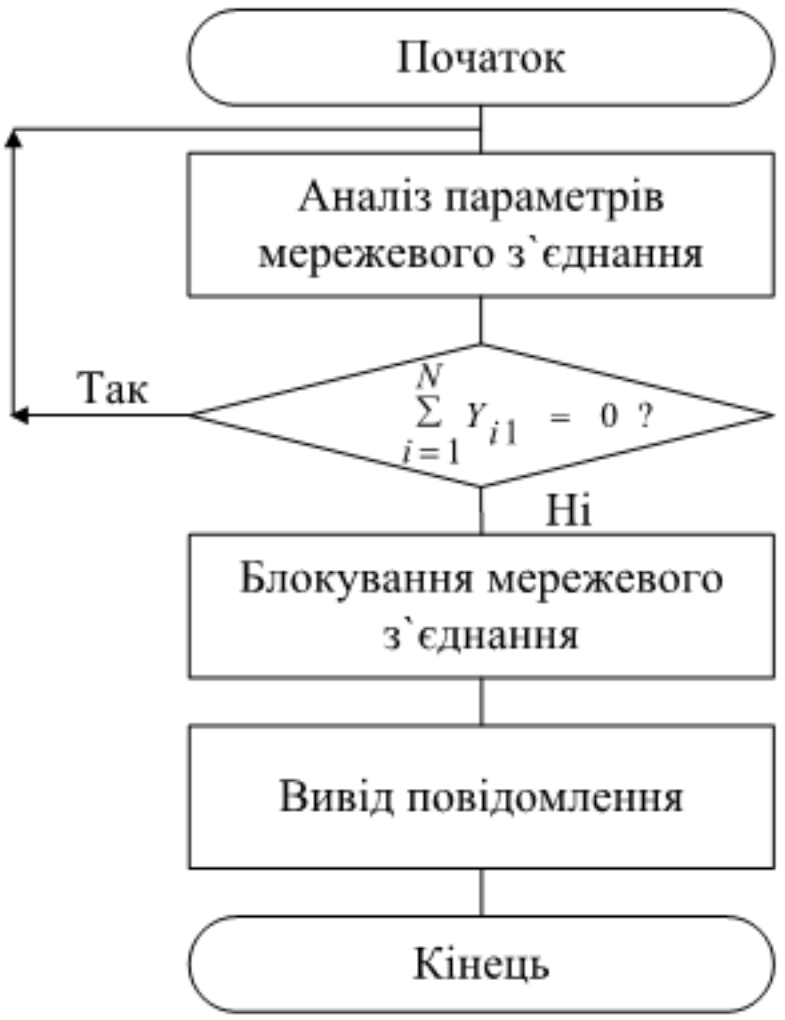


Рисунок 4.5 – Схема алгоритму роботи виявлення і класифікації мережевих атак

Як наголошувалося в розділі 3, кожний НІД має так званий «час життя», впродовж якого він може аналізувати мережевий трафік. Такий обмежений проміжок існування детекторів необхідний для того, щоб позбутися від «слабких» детекторів. Справа в тому, що після навчання і відбору нейромережевих детекторів немає гарантії, що детектор здатний виявляти і класифікувати мережеві атаки. Точніше сказати, може виникнути така ситуація, коли детектор класифікуватиме невідомий образ, як атаку, тільки у тому випадку, коли цей образ в точності співпадатиме з навчальною вибіркою. Така ситуація може виникнути через те, що навчальна вибірка для кожного детектора формується випадковим чином, і можлива така ситуація, коли нейронна мережа, що лежить в основі детектора, на навчальних даних

не зможе виявити закономірності в параметрах з'єднань, що відносяться до різних типів і класів. Механізм обмеження часу функціонування, коли детектор знищується, якщо він не виявив протягом заданого часу атаку, дозволяє позбутися від «слабких» детекторів. При знищенні детектора, на його місце приходить новий НІД, який прошов стадії навчання і відбору.

Сучасна система захисту комп'ютерів від мережевих атак повинна не тільки надійно захищати від вже відомих мережевих атак, але і також від невідомих, таких, що раніше не зустрічалися. Тобто система повинна мати властивості самоадаптації до зміни «сигнатур» мережевих атак і методів їх організацій та здійснення. За самоадаптацію в розробленій і пропонованій системі виявлення і класифікації мережевих атак відповідає підсистема адаптації [56].

Модуль адаптації заснований на дослідженні характеристик виявленої мережевої атаки і здатності НІД донавчатися.

Опишемо в деталях розроблений механізм адаптації. При виявленні мережевої атаки одним з детекторів, відбувається блокування даного мережевого з'єднання і генерується повідомлення користувача. Проте, окрім вказаних дій, система запам'ятовує характеристики мережевого з'єднання, класифікованого як атака. Далі, параметри даної атаки порівнюються з параметрами атак, що знаходяться в базі даних для навчання детекторів. Якщо атака з такими або достатньо близькими характеристиками вже існує, то нічого не відбувається. Проте, якщо атаки з такими характеристиками в базі не існує, або характеристики виявленої атаки досить сильно відрізняються від вже відомих, то проводяться наступні операції:

1. На основі нейромережевого детектора, який виявив мережеву атаку, створюється новий детектор, так званий детектор імунної пам'яті. На першому етапі це просте дублювання детектора (операція клонування).

2. Проводиться донавчання нового детектора на параметрах нової виявленої мережевої атаки (операція мутації).

3. Новий детектор, що пройшов донавчання, впроваджується в

підсистему виявлення і класифікації мережевих атак.

4. Параметри нової мережевої атаки заносяться в базу, що зберігає дані для навчання НІД.

У загальному вигляді схема алгоритму роботи модуля адаптації представлена на рисунку 4.6 [56].

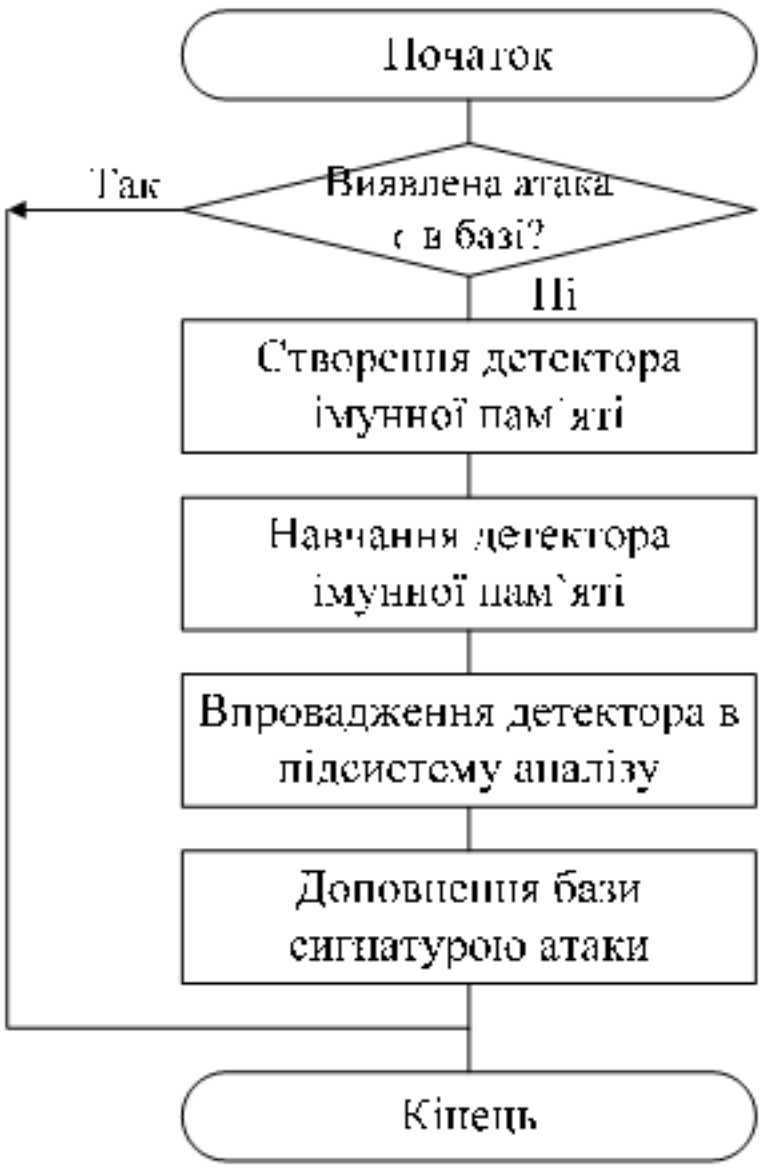


Рисунок 4.6 – Схема алгоритму роботи модуля адаптації

Описаний алгоритм дозволяє всій системі виявлення і класифікації мережевих вторгнень аналізувати виявлену мережеву атаку і, якщо виявляється її унікальність, тобто атака є новою, раніше невідомою, то адаптуватися до неї шляхом створення детекторів імунної пам’яті і внесенням характеристик (сигнатури) нової мережевої атаки до навчальної вибірки для навчання нових детекторів.

Для усунення конфліктів в роботі нейромережевих детекторів, коли одну атаку виявляють і класифікують декілька детекторів, використовується підхід, запропонований в параграфі 2.4.2.

Отже, в даному підрозділі представлено структуру програмного забезпечення та алгоритми функціонування нейромережевої імунної системи виявлення і класифікації атак на ІТМ.

4.2 Реалізація програмного забезпечення

Для програмної реалізації використано мову програмування C++, а засобом реалізації вибрано середовище програмування C++ Builder 7.0.

При запуску програмного модуля з'являється головне вікно з інформацією про програмний продукт і кнопками вибору: запуск програми або вихід з програми (рисунок 4.7).

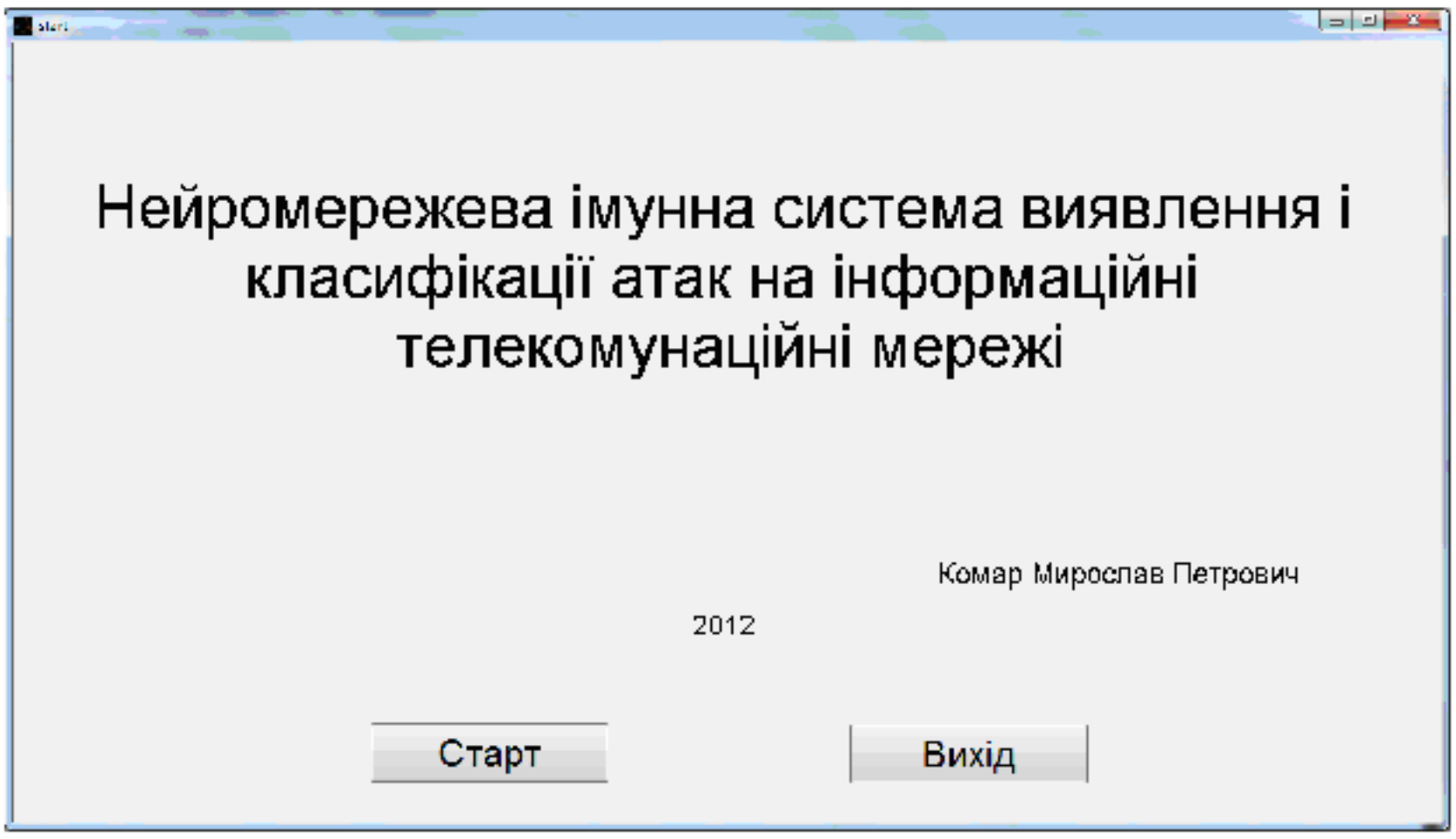


Рисунок 4.7 – Головне вікно програмного забезпечення

При натисненні на кнопку «Старт» відбувається запуск основних компонентів програмного модуля і система переходить в робочий стан. При натисненні на кнопку «Вихід» відбувається закриття головного вікна і завершення роботи програми.

Інтерфейс програми побудований таким чином, що можна перемикатися між двома основними інтерфейсами: користувацьким і

режимом налаштування системи.

Режим користувацького інтерфейсу є режимом за замовчуванням. При роботі програми в даному режимі у вікні відображається вся інформація про поточний стан системи аналізу мережевого трафіка, про поточні мережеві з'єднання, кількість виявлених мережевих атак. Також в даному вікні є кнопки для формування звіту, зупинки аналізу трафіку, завершення роботи програми, переходу в режим налаштувань системи.

Вікно користувацького інтерфейсу відображене на рисунку 4.8.

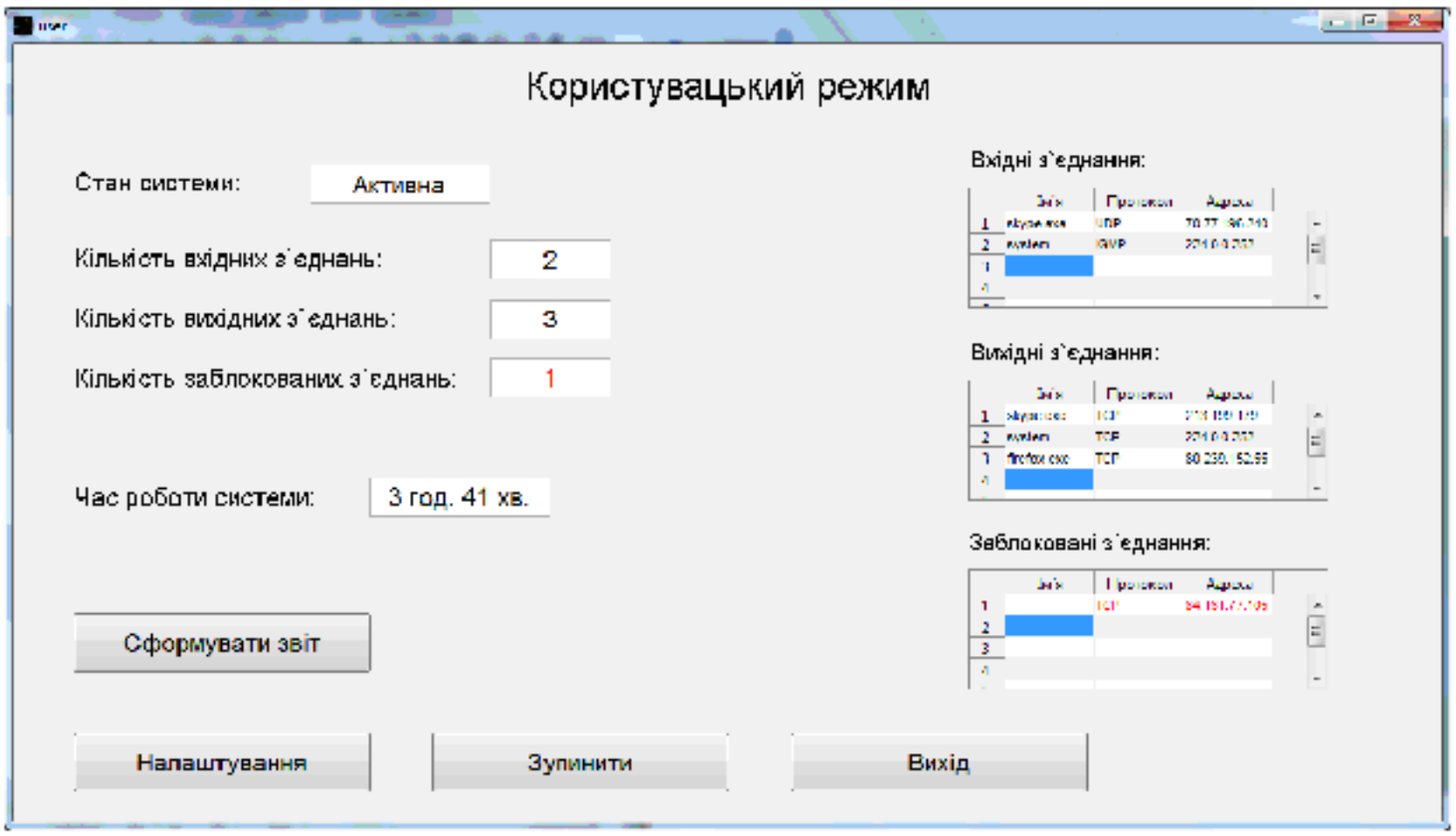


Рисунок 4.8 – Вікно користувацького інтерфейсу

Розглянемо детально інтерфейс користувацького режиму. У полі «Стан системи» видається інформація про активність або неактивність системи аналізу мережевих з'єднань. У стані «Активна» відбувається аналіз вхідних і вихідних мережевих з'єднань, тобто система працює в штатному режимі. При натисненні кнопки «Зупинити» система переходить в неактивний стан. У цьому випадку аналіз мережевого трафіку припиняється.

Нижче видається інформація про кількість активних вхідних і вихідних мережевих з'єднань, а також про кількість заблокованих з'єднань за поточний сеанс роботи. Далі видається інформація про час роботи системи в

поточному сеансі.

У правій частині вікна розташовуються три таблиці, в яких відображається інформація (ім'я процесу, якщо таке є, протокол передачі і мережева адреса) про активні вхідні і вихідні мережеві з'єднання, а також про заблоковані з'єднання.

Ряд кнопок у вікні користувацького режиму служать для управління системою. Так, при натисканні кнопки «Зупинити» відбувається припинення аналізу мережевого трафіку і система переходить в стан очікування. Натиснення кнопки «Вихід» приведе до завершення роботи програми.

При натисненні кнопки «Сформувати звіт» створюється нове вікно (рисунок 4.9), в якому відображається інформація про всі значущі події за період часу, що дорівнює 7 днів.

	Програма	Дія	Протокол	IP адреса	Тип атаки	Дата
1	Lovesan.exe	Заблоковано	TCP	192.168.1.10	Warezclient	02.04.2012
2	lsass.exe	Заблоковано	TCP	192.168.1.10	Spy	02.04.2012
3		Заблоковано	TCP	192.168.1.10	DoS	29.03.2012
4		Заблоковано	UDP	192.168.1.10	Unknown	28.03.2012
5						
6						
7						
8						
9						
10						
11						
12						

Рисунок 4.9 – Звіт системи аналізу трафіку

При натисненні кнопки «Налаштування» відбувається перемикання режимів інтерфейсу. Режим налаштування системи (рисунок 4.10) призначений для адміністрування і налаштування багатьох параметрів системи. Розглянемо даний режим детально.

У блоці «Детектори» можна стежити за кожним НІД. Тут видається інформація про кількість в системі активних НІД, які в даний момент

аналізують мережевий трафік. Також видається інформація про кількість детекторів імунної пам'яті, тобто тих НІД, які виявляли мережеві атаки.

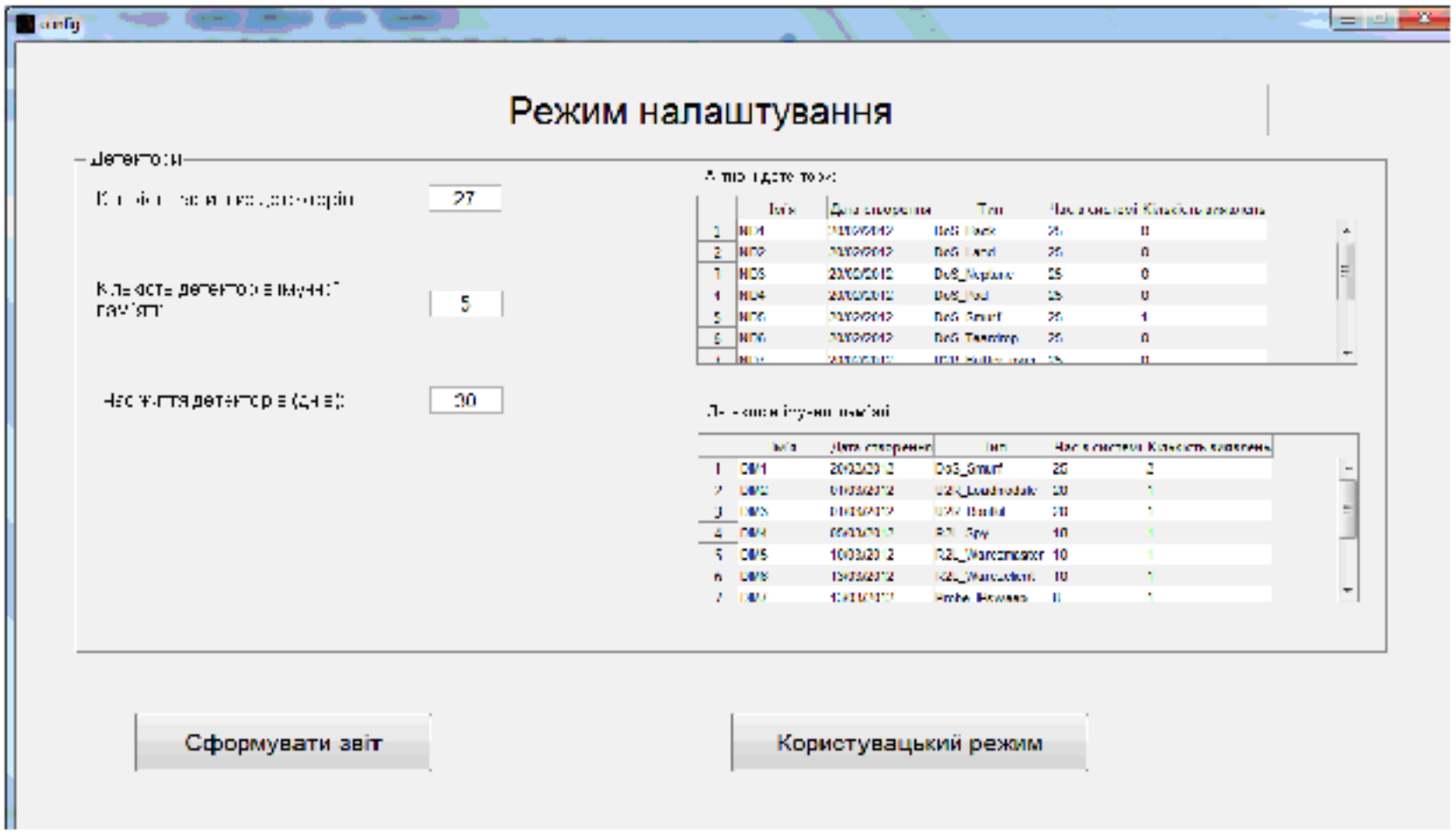


Рисунок 4.10 – Режим налаштувань системи

У правій частині блоку видається детальна інформація про кожен детектор. У таблиці НІД відображається наступна інформація: ім'я детектора, дата створення детектора, тип детектора (тип детектора вказує на те, які атаки навчений виявляти даний детектор), час перебування детектора в системі, кількість виявлених атак.

Нижче, також в табличній формі, відображається детальна інформація про детектори імунної пам'яті: ім'я детектора, дата створення детектора, тип детектора, ім'я батьківського детектора, час перебування детектора в системі і кількість виявлених атак даним детектором.

У нижній частині екрану розташовуються кнопки формування звіту і перемикання в користувацький режим. При натисненні кнопки «Сформувати звіт» відбувається вивід і збереження інформації з описаних вище таблиць. А при натисненні кнопки «Користувацький режим» відбувається повернення в користувацький режим інтерфейсу системи.

Фрагменти лістингів програмного забезпечення подано в додатку Л.

У даному підрозділі представлено програмну реалізацію нейромережевої імунної системи виявлення і класифікації атак на ІТМ.

4.3 Статистична оцінка достовірності розробленої інтелектуальної інформаційної технології

З метою усунення недоліків відомих рішень, підвищення достовірності виявлення і класифікації атак на ІТМ розроблено інтелектуальну інформаційну технологію (ІІТ) шляхом використання методів штучного інтелекту, а саме методів штучних нейронних мереж і штучних імунних систем [53, 56] (рисунок 4.11).

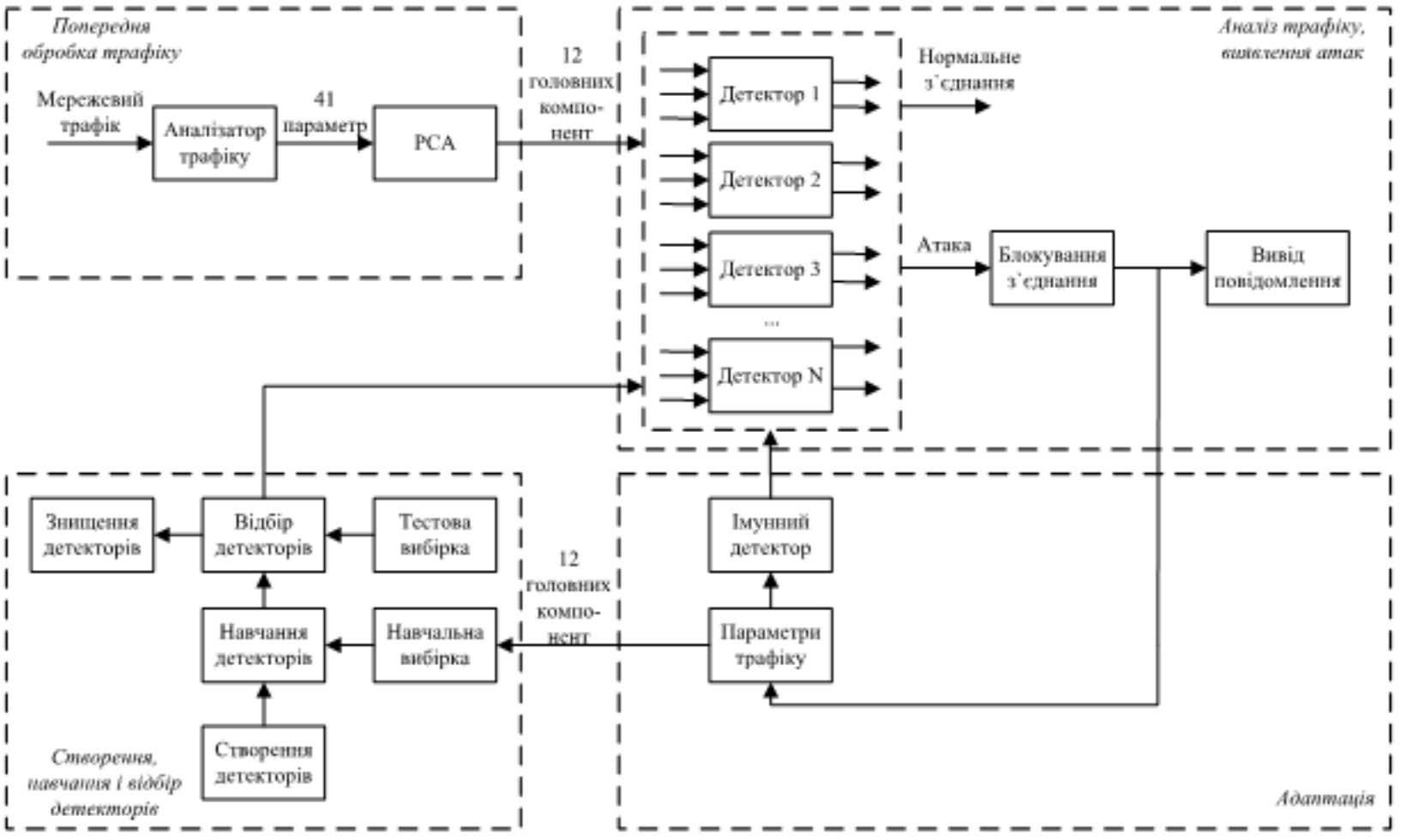


Рисунок 4.11 – Схема інтелектуальної інформаційної технології виявлення і класифікації атак на ІТМ

Проведемо статистичну оцінку достовірності ІІТ виявлення і класифікації атак на ІТМ на основі ROC-аналізу і, відповідно, критеріїв розглянутих в підрозділі 1.3 [63].

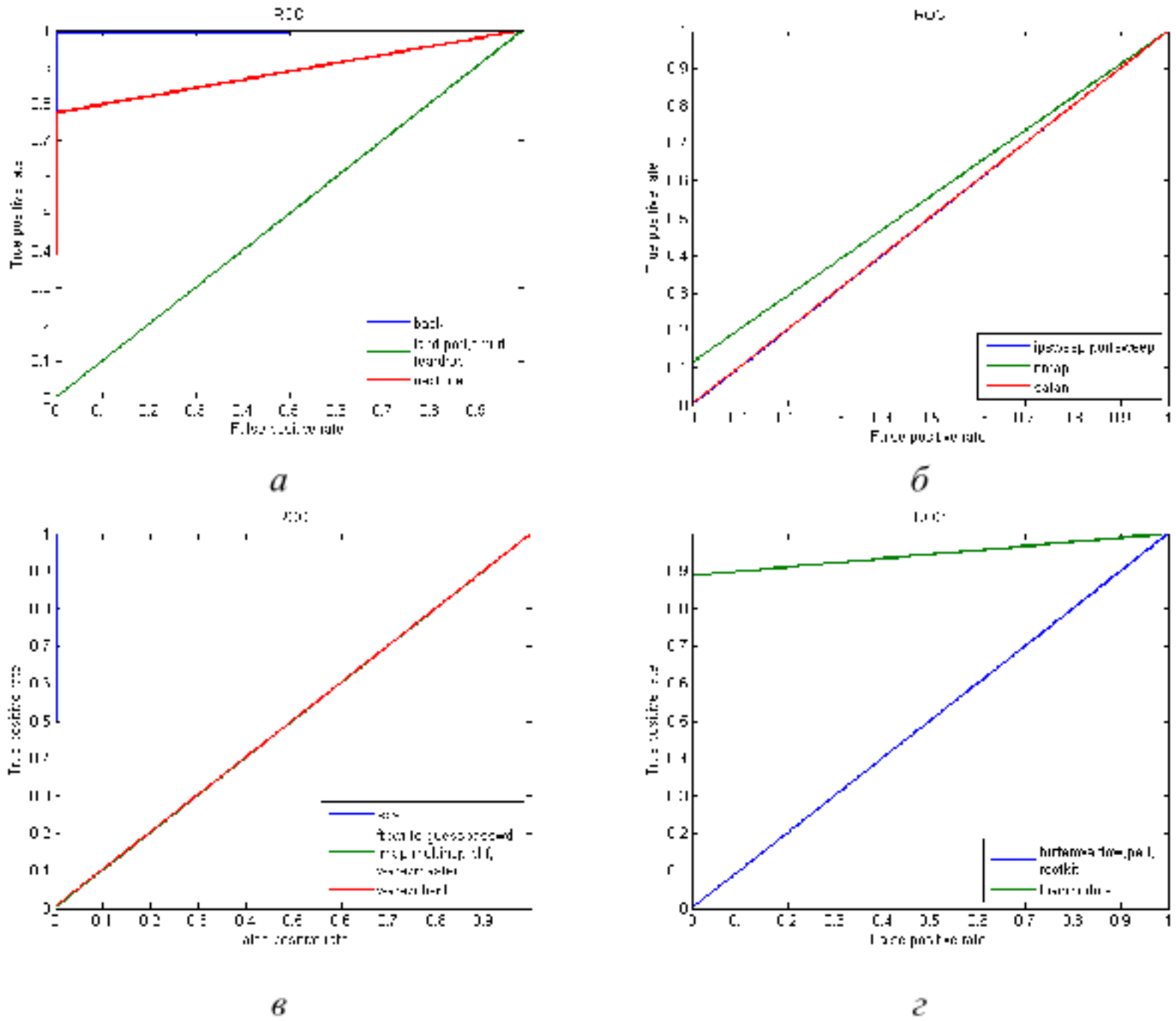
У таблицях 4.3–4.8 представлені характеристики шести НІД. А на

рисунках 4.12–4.17 зображені відповідно їх ROC-криві, що відображають властивість детекторів виявляти і класифікувати мережеві атаки класів *DoS*, *Probe*, *R2L* і *U2R*.

Таблиця 4.3 – Характеристики НІД 1

Детектор 1: навчання на *dos_back*. Sp (TNR) = 99,8%, FPR = 0,2%			
Тип атаки	*Se (TPR),%*	*FNR,%*	*Accu,%*
DoS-атаки			
back	100,0	0,0	99,9
land	0,0	100,0	50,0
neptune	84,9	15,1	92,3
pod	0,0	100,0	50,0
smurf	0,0	100,0	50,0
teardrop	0,0	100,0	50,0
Probe-атаки			
ipsweep	0,0	100,0	50,0
nmap	11,7	88,3	55,8
portsweep	0,0	100,0	50,0
satan	0,5	99,5	50,2
R2L-атаки			
ftp_write	0,0	100,0	50,0
guess_passwd	0,0	100,0	50,0
imap	0,0	100,0	50,0
multihop	0,0	100,0	50,0
phf	0,0	100,0	50,0
spy	100,0	0,0	99,9
warezclient	0,5	99,5	50,2
warezmaster	0,0	100,0	50,0
U2R-атаки			
buffer_overflow	0,0	100,0	50,0
loadmodule	88,9	11,1	94,4
perl	0,0	100,0	50,0
rootkit	0,0	100,0	50,0

Як видно з таблиці 4.3 і рисунку 4.12, детектор 1 досить добре виявляє тип атак *dos_back*, на якому і проходило його навчання. Він виявив 100% атак даного типу при низькому рівні помилок другого роду FPR = 0,2%. Окрім цього, даний детектор також добре виявляє атаки класу *dos_neptune* – 84,9%, атаки класу *r2l_spy* – 100% і атаки класу *u2r_loadmodule* – 88,9%.



а *б*

в *г*

Рисунок 4.12 – ROC-криві виявлення і класифікації атак НІД 1:

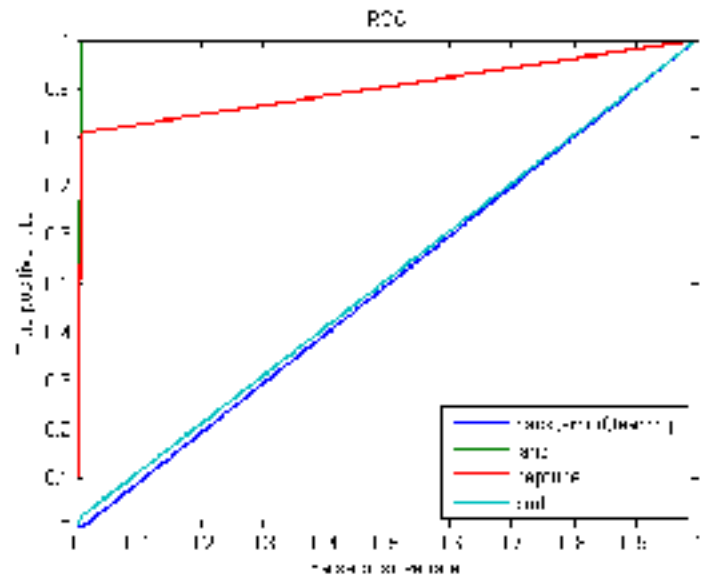
а – DoS-атак, *б – Probe*-атак, *в – R2L*-атак, *г – U2R*-атак

Таблиця 4.4 – Характеристики НІД 2

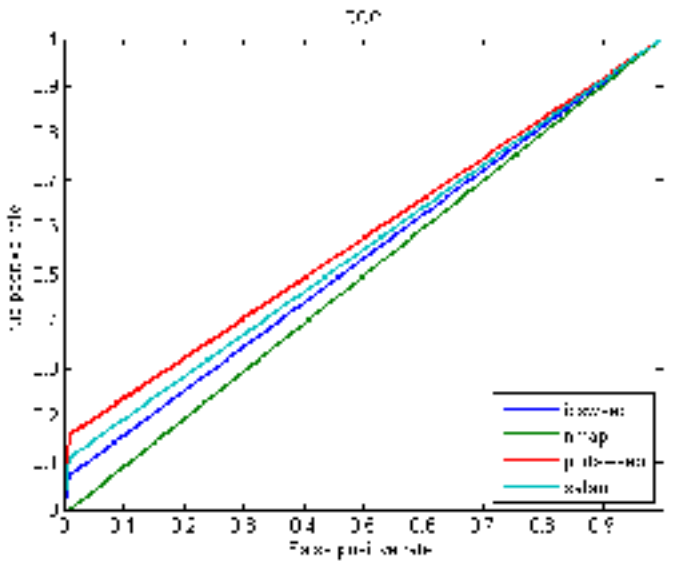
Детектор 2: навчання на *dos_land*. Sp (TNR) = 99,0%, FPR = 1,0%			
Тип атаки	*Se (TPR),%*	*FNR,%*	*Accu,%*
DoS-атаки			
Back	0,0	100,0	50,0
Land	100,0	0,0	99,5
Neptune	88,5	11,5	93,8
Pod	2,3	97,7	50,6
Smurf	0,0	100,0	50,0
Teardrop	0,0	100,0	50,0
Probe-атаки			
Ipsweep	7,22	92,8	53,1
Nmap	0,0	100,0	50,0
Portsweep	15,8	84,2	57,4
Satan	11,0	89,0	55,0

Продовження таблиці 4.4

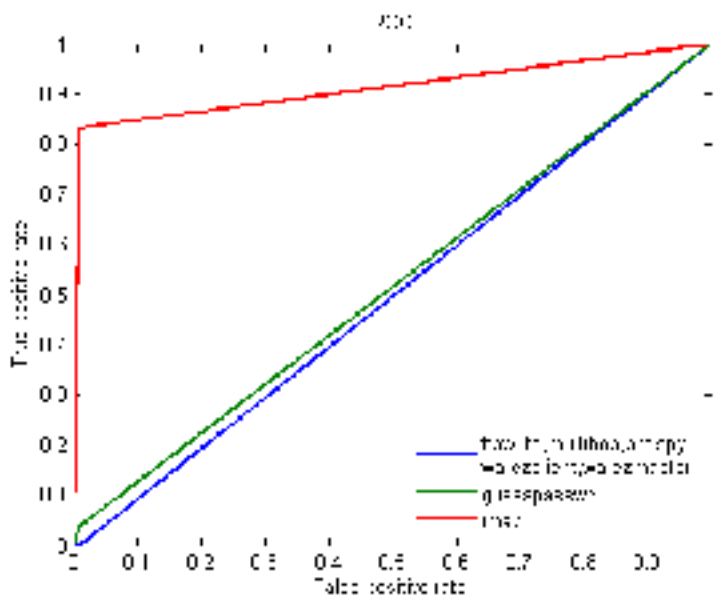
R2L-атаки			
ftp_write	0,0	100,0	50,0
guess_passwd	3,8	96,2	51,4
Imap	83,3	16,7	91,2
Multihop	0,0	100,0	50,0
Phf	0,0	100,0	50,0
Spy	0,0	100,0	50,0
Warezclient	0,2	99,8	0,50
Warezmaster	0,0	100,0	50,0
U2R-атаки			
buffer_overflow	0,0	100,0	50,0
Loadmodule	0,0	100,0	50,0
Perl	0,0	100,0	50,0
Rootkit	0,0	100,0	50,0



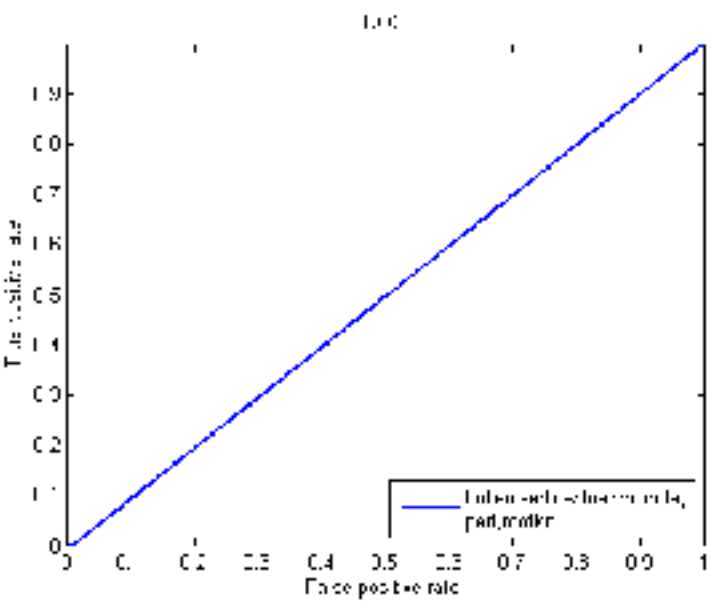
а



б



в



г

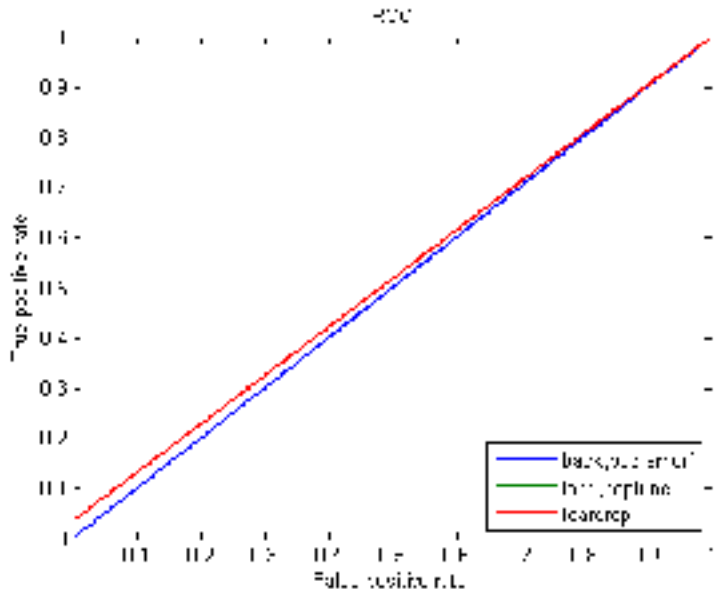
Рисунок 4.13 – ROC-криві виявлення і класифікації атак НІД 2:

а – DoS-атак, *б – Probe*-атак, *в – R2L*-атак, *г – U2R*-атак

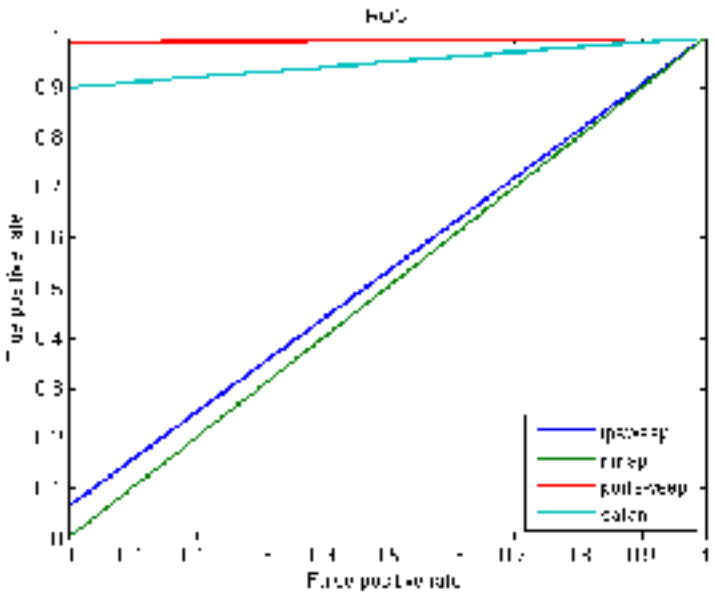
Як видно з таблиці 4.4 і рисунку 4.13 детектор 2 із стовідсотковою ймовірністю виявляє атаки *dos_land*, на яких проходило його навчання. При цьому рівень помилок другого рівня FPR = 1,0%. Також даний детектор виявив атаки *dos_neptune* з ймовірністю рівною 88,5% і атаки *r2l_imap* з ймовірністю 83,3%.

Таблиця 4.5 – Характеристики НІД 3

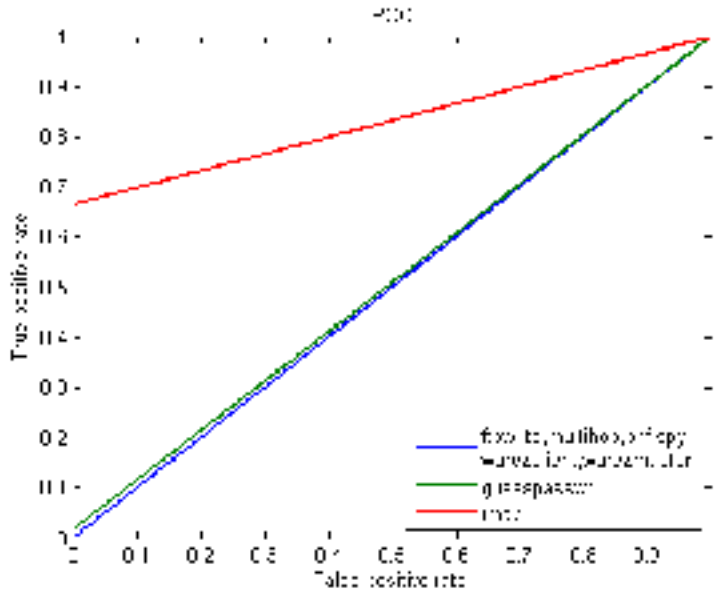
Детектор 3: навчання на *dos_neptune*. Sp (TNR) = 99,9%, FPR = 0,1%			
Тип атаки	*Se (TPR),%*	*FNR,%*	*Accu,%*
DoS-атаки			
Back	0,0	100,0	50,0
land	100,0	0,0	100,0
neptune	100,0	0,0	100,0
pod	0,0	100,0	50,0
smurf	0,0	100,0	50,0
teardrop	3,7	96,3	51,8
Probe-атаки			
ipsweep	6,5	93,5	53,2
nmap	0,0	100,0	50,0
portsweep	98,9	1,1	99,4
satan	90,0	10,0	95,0
R2L-атаки			
ftp_write	0,0	100,0	50,0
guess_passwd	1,9	98,1	50,1
imap	66,7	33,3	83,3
multihop	0,0	100,0	50,0
phf	0,0	100,0	50,0
spy	0,0	100,0	50,0
warezclient	0,2	99,8	50,1
warezmaster	0,0	100,0	50,0
U2R-атаки			
buffer_overflow	0,0	100,0	50,0
loadmodule	0,0	100,0	50,0
perl	0,0	100,0	50,0
rootkit	0,0	100,0	50,0



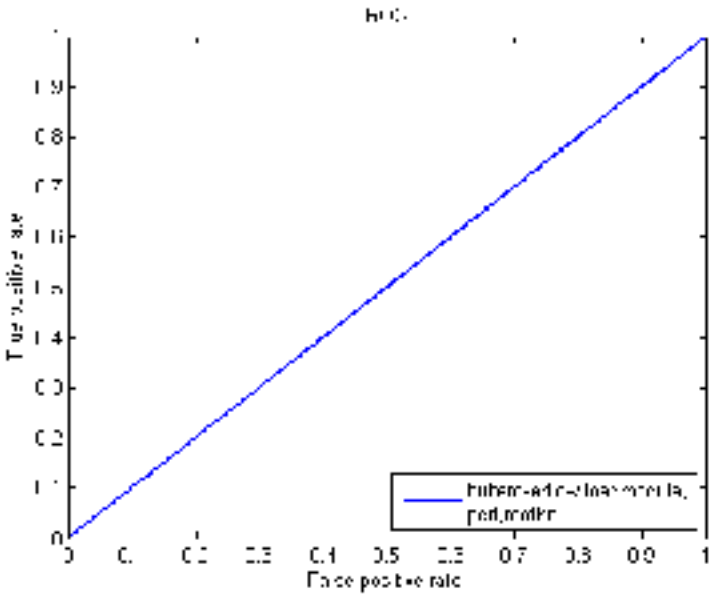
а



б



в



г

Рисунок 4.14 – ROC-криві виявлення і класифікації атак НІД 3:
а – DoS-атак, *б – Probe*-атак, *в – R2L*-атак, *г – U2R*-атак

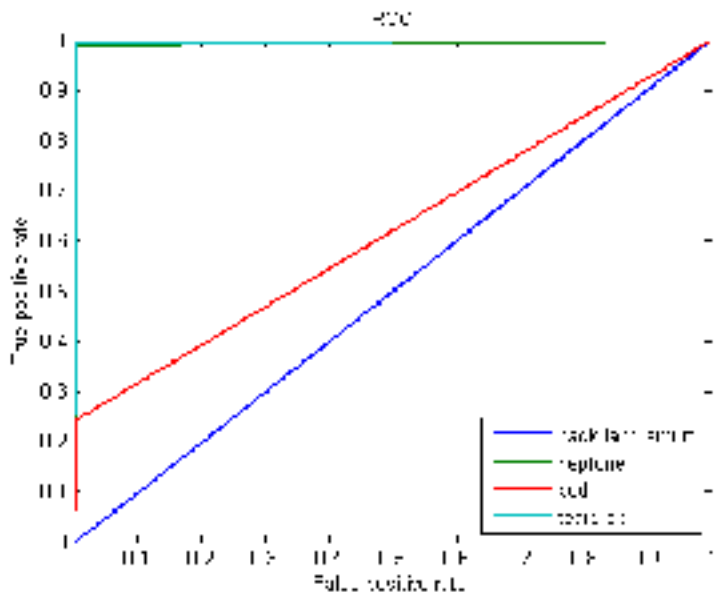
Детектор 3 навчався на атаках *dos_neptune*. Він виявляє атаки *dos_land* і *dos_neptune* з ймовірністю рівною 100% при рівні помилок другого роду FPR = 0,1%.

Окрім цього, детектор 3 з різною ймовірністю виявляє наступні типи атак: *probe_portsweep* – 98,9%, *probe_satan* – 90,0%, *r2l_imap* – 66,7%.

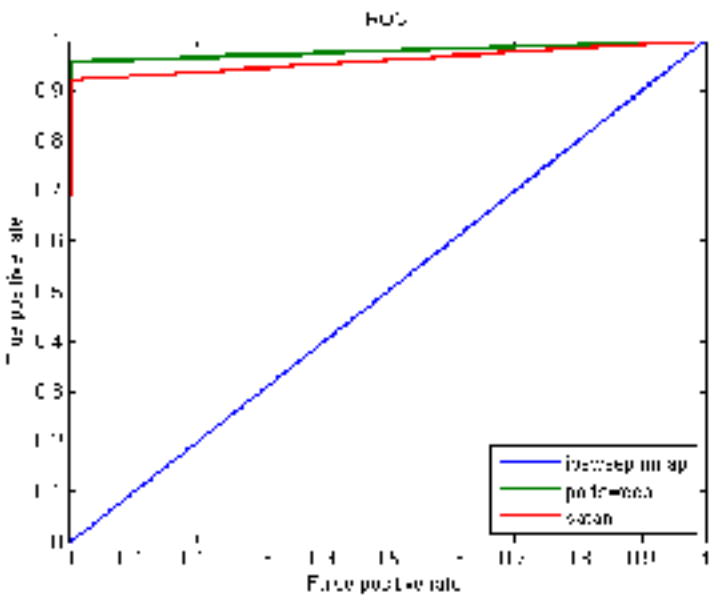
Таблиця 4.6 – Характеристики НІД 4

Детектор 4: навчання на *dos_teardrop*. Sp (TNR) = 99,6%, FPR = 0,4%			
Тип атаки	*Se (TPR),%*	*FNR,%*	*Accu,%*
DoS-атаки			
back	0,0	100,0	50,0
land	0,0	100,0	50,0
neptune	98,9	1,1	99,3
pod	24,2	75,8	61,9
smurf	0,0	100,0	50,0
teardrop	99,8	0,02	99,7
Probe-атаки			
ipsweep	0,0	100,0	50,0
nmap	0,0	100,0	50,0
portsweep	95,7	4,3	97,6
satan	92,0	8,0	95,8
R2L-атаки			
ftp_write	0,0	100,0	50,0
guess_passwd	0,0	100,0	50,0
imap	0,0	100,0	50,0
multihop	0,0	100,0	50,0
phf	0,0	100,0	50,0
spy	0,0	100,0	50,0
warezclient	0,0	100,0	50,0
warezmaster	0,0	100,0	50,0
U2R-атаки			
buffer_overflow	0,0	100,0	50,0
loadmodule	0,0	100,0	50,0
perl	0,0	100,0	50,0
rootkit	0,0	100,0	50,0

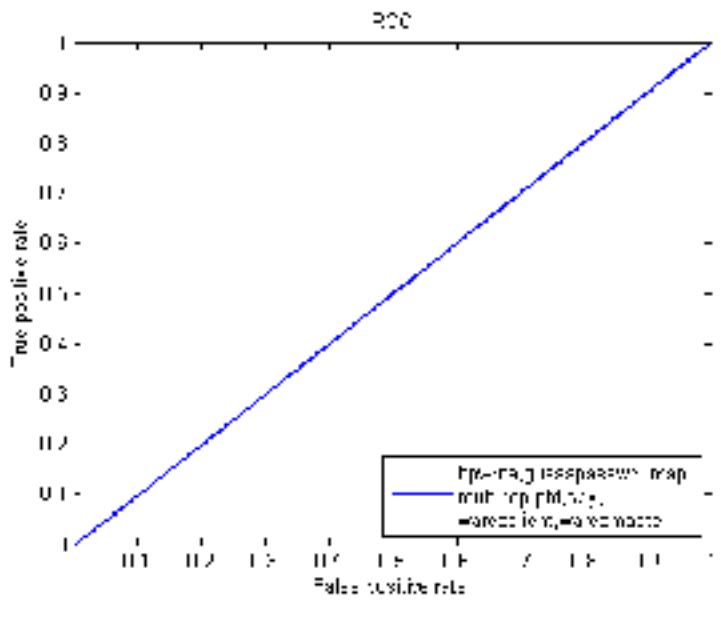
Як видно з таблиці 4.6 і рисунку 4.15, детектор 4, навчаючись на атаках класу *dos_teardrop*, виявляє їх з ймовірністю, рівною 99,8%, при рівні помилок другого роду рівної FPR = 0,4%. Окрім цього, він виявив атаки класу *dos_neptune* з ймовірністю, рівною 98,9%, атаки класу *dos_pod* з ймовірністю, рівною 24,2%, атаки класів *probe_portsweep* і *probe_satan* з ймовірністю, рівною 95,7% і 92,0% відповідно.



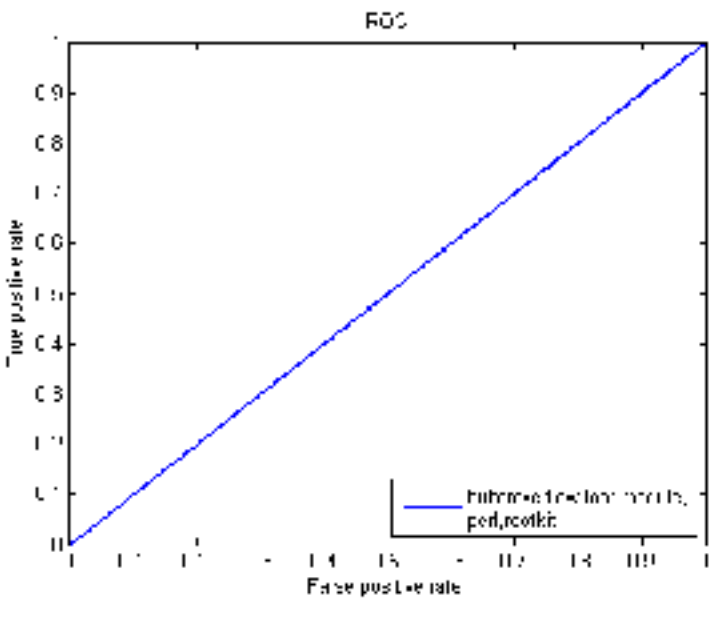
а



б



в



г

Рисунок 4.15 – ROC-криві виявлення і класифікації атак НІД 4:

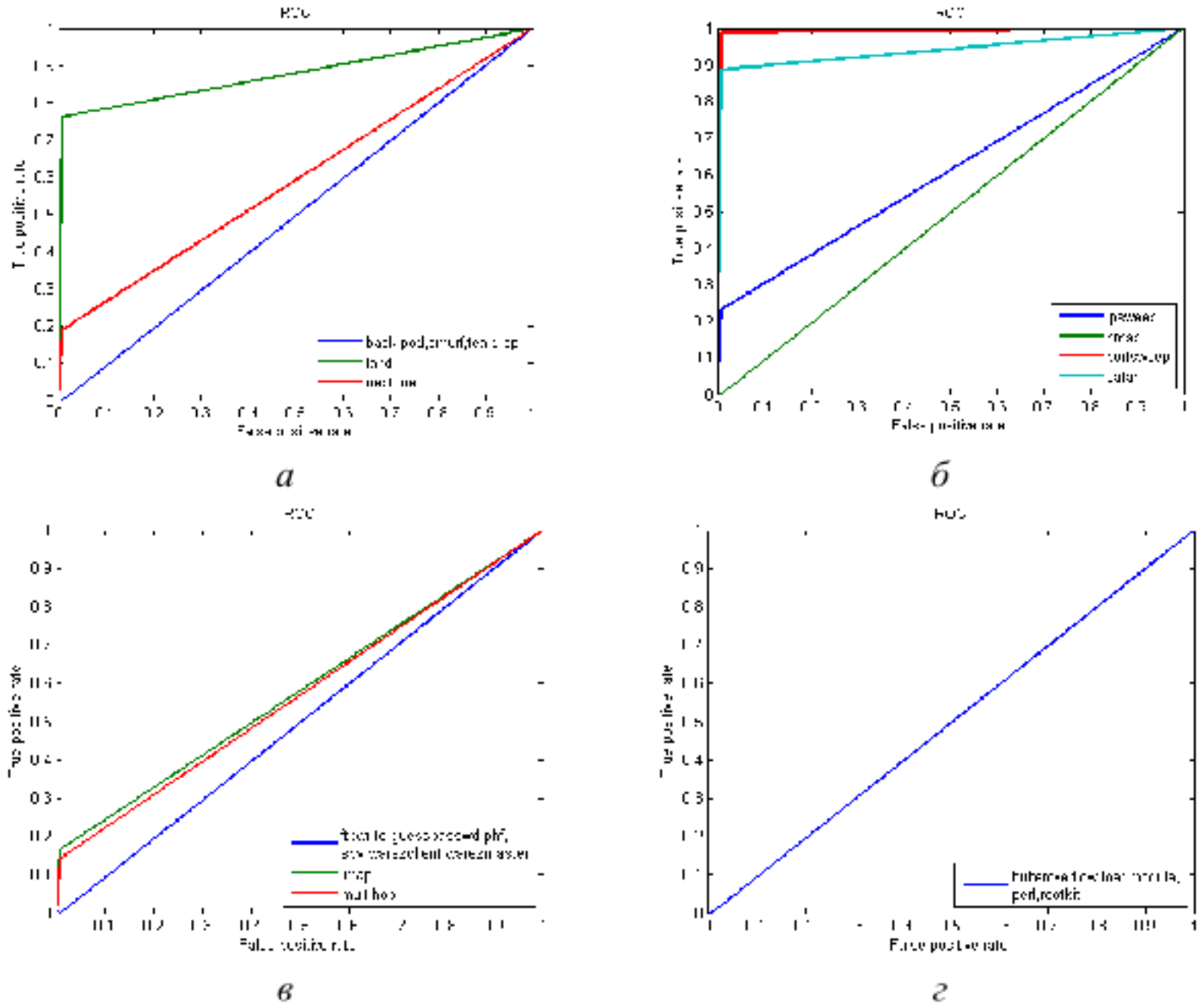
а – *DoS*-атак, *б* – *Probe*-атак, *в* – *R2L*-атак, *г* – *U2R*-атак

Таблиця 4.7 – Характеристики НІД 5

Детектор 5: навчання на *Probe_portsweep*. Sp (TNR) = 99,2%, FPR = 0,8%			
Тип атаки	*Se (TPR),%*	*FNR,%*	*Accu,%*
DoS-атаки			
back	0,0	100,0	50,0
land	76,2	23,8	87,7
neptune	11,5	88,5	55,3
pod	0,0	100,0	50,0
smurf	0,0	100,0	50,0
teardrop	0,0	100,0	50,0
Probe-атаки			
ipsweep	23,2	76,8	61,2
nmap	0,0	100,0	50,0
portsweep	98,9	1,1	99,1
satan	88,8	11,2	94,0

Продовження таблиці 4.7

R2L-атаки			
ftp_write	0,0	100,0	50,0
guess_passwd	0,0	100,0	50,0
imap	16,7	83,3	57,9
multihop	14,3	85,7	65,7
Phf	0,0	100,0	50,0
spy	0,0	100,0	50,0
warezclient	0,0	100,0	50,0
warezmaster	0,0	100,0	50,0
U2R-атаки			
buffer_overflow	0,0	100,0	50,0
loadmodule	0,0	100,0	50,0
perl	0,0	100,0	50,0
rootkit	0,0	100,0	50,0



а *б* *в* *г*

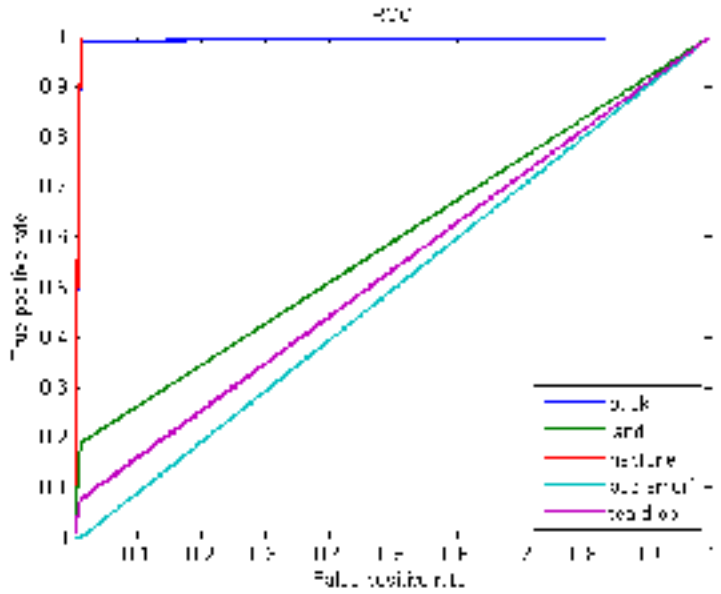
Рисунок 4.16 – ROC-криві виявлення і класифікації атак НІД 5:

а – DoS-атак, *б – Probe*-атак, *в – R2L*-атак, *г – U2R*-атак

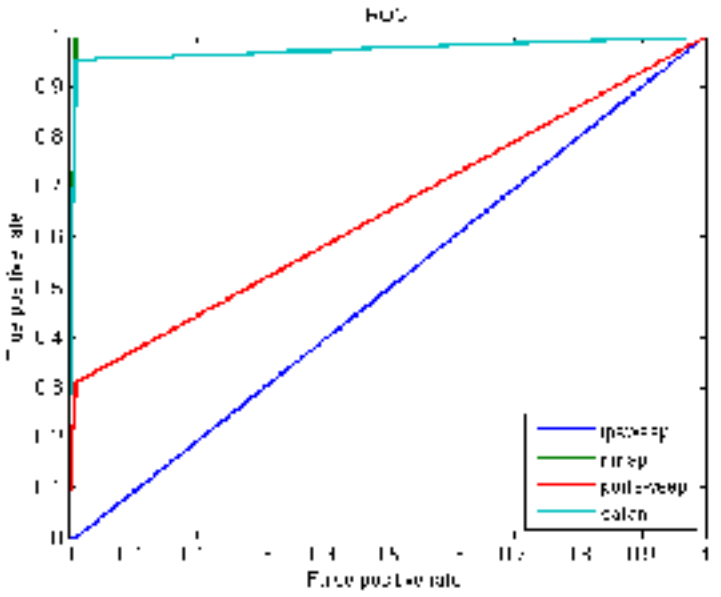
З таблиці 4.7 і рисунку 4.16 видно, що детектор 5, який навчався на атаках класу *probe_portsweep*, виявляє їх з ймовірністю, рівною 98,9%, при рівні помилок другого роду FPR = 0,8%. Окрім цього, він також виявляє наступні атаки: *dos_land* – 76,2%, *dos_neptune* – 11,5%, *probe_ipsweep* – 23,2%, *probe_satan* – 88,8%, *r2l_imap* – 16,7%, *r2l_multihop* – 14,3%.

Таблиця 4.8 – Характеристики НІД 6

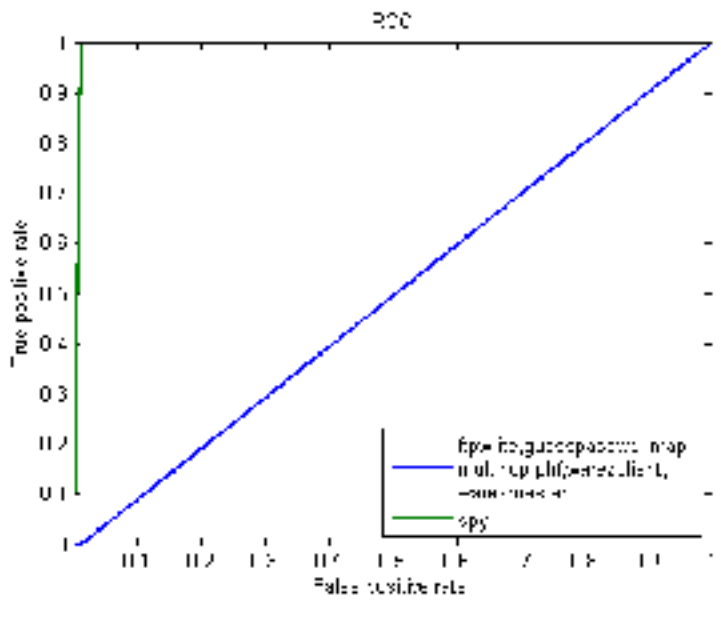
Детектор 6: навчання на *Probe_nmap*. Sp (TNR) = 98,8%, FPR = 1,2%			
Тип атаки	*Se (TPR),%*	*FNR,%*	*Accu,%*
DoS-атаки			
Back	99,0	1,0	98,9
land	19,1	80,9	58,9
neptune	99,9	0,1	99,4
pod	0,0	100,0	50,0
smurf	0,0	100,0	50,0
teardrop	7,6	92,4	53,2
Probe-атаки			
ipsweep	0,0	100,0	50,0
nmap	100,0	0,0	99,4
portsweep	30,9	69,1	64,9
satan	95,2	4,8	97,0
R2L-атаки			
ftp_write	0,0	100,0	50,0
guess_passwd	0,0	100,0	50,0
imap	0,0	100,0	50,0
multihop	0,0	100,0	50,0
phf	0,0	100,0	50,0
spy	100,0	0,0	99,4
warezclient	0,1	99,9	50,0
warezmaster	0,0	100,0	50,0
U2R-атаки			
buffer_overflow	0,0	100,0	50,0
loadmodule	100,0	0,0	99,4
perl	0,0	100,0	50,0
rootkit	0,0	100,0	50,0



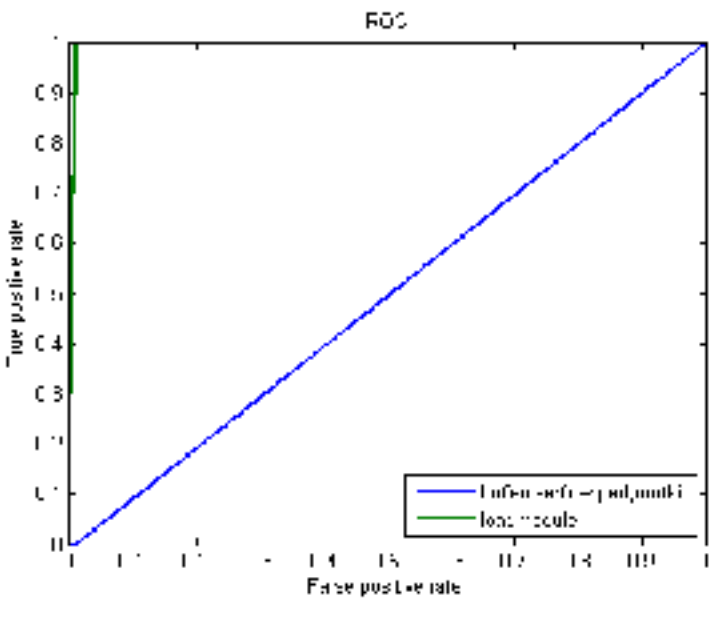
а



б



в



г

Рисунок 4.17 – ROC-криві виявлення і класифікації атак НІД 6:
а – *DoS*-атак, *б* – *Probe*-атак, *в* – *R2L*-атак, *г* – *U2R*-атак

Детектор 6 навчався на атаках класу *probe_nmap*. Він показав наступні результати: величина помилки другого роду – 1,2%, ймовірність виявлення атак *dos_back* – 99,0%, *dos_land* – 19,1%, *dos_neptune* – 99,9%, *dos_teardrop* – 7,6%, *probe_nmap* – 100%, *probe_portsweep* – 30,9%, *probe_satan* – 95,2%, *r2l_spy* – 100%, *u2r_loadmodule* – 100%.

Представлені результати експериментальних досліджень доводять здатність НІД коректно і точно виявляти та класифікувати мережеві атаки при низькому рівні виникнення помилок другого роду. Детектори, навчаючись на певному типі мережевих атак, виявляють даний тип атак з ймовірністю, близькою до 100%. Окрім цього, різні детектори здатні з

високою ймовірністю виявляти й деякі інші типи атак, навчання на яких не проводилося. Тобто НІД показують здатність виявляти невідомі атаки, що є важливою властивістю сучасних систем виявлення вторгнень. Таким чином, популяція НІД, де кожен детектор навчений на окремому типі атак, забезпечує надійне виявлення відомих, а також і невідомих мережевих атак.

У таблиці 4.9 представлені результати виявлення і класифікації мережевих атак на основі 10 НІД.

Таблиця 4.9 – Достовірність виявлення і класифікації мережевих атак

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
d_back	99,05	0,00	0,00	0,00	0,00	0,00	99,05	0,00	0,00	0,27
d_land	0,00	100	100	61,91	4,76	80,95	9,52	100	4,76	23,81
d_neptune	99,07	80,91	100	0,00	0,00	99,99	99,85	100	99,95	0,00
d_pod	0,00	2,27	0,00	0,34	0,00	31,06	12,88	0,00	0,00	99,89
d_smurf	0,00	0,00	0,00	99,92	100	0,14	0,03	0,00	0,00	0,00
d_teardrop	0,00	0,00	2,66	0,00	32,99	100	10,93	8,79	7,66	0,00
p_ipsweep	99,08	7,22	6,58	65,20	45,07	6,98	5,69	6,98	6,74	0,96
p_nmap	80,52	0,00	0,00	0,00	0,00	96,10	100	0,00	0,00	0,00
p_portsweep	2,02	15,88	98,85	0,48	1,44	97,79	30,80	99,90	98,56	0,10
p_satan	13,28	11,01	88,80	0,00	13,59	93,90	96,04	92,20	94,59	2,08
r_ftpwrite	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100
r_gpasswd	0,00	3,77	94,34	0,00	0,00	0,00	0,00	1,89	0,00	5,66
r_imap	0,00	83,33	83,33	0,00	99,10	16,67	0,00	75,00	0,00	0,00
r_multihop	0,00	0,00	0,00	97,60	0,00	0,00	0,00	0,00	0,00	57,14
r_phf	0,00	98,70	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
r_spy	100	0,00	0,00	0,00	0,00	50,00	100	0,00	0,00	0,00
r_wclient	1,08	0,20	0,20	90,00	0,00	0,00	0,59	0,20	0,00	65,00
r_wmaster	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	90,00
u_overflow	0,00	0,00	0,00	0,00	99,00	0,00	0,00	3,33	0,00	83,33
u_loadmodule	88,89	0,00	0,00	0,00	0,00	11,11	100	0,00	0,00	0,00
u_perl	33,33	0,00	0,00	0,00	0,00	97,00	0,00	0,00	0,00	0,00
u_rootkit	0,00	0,00	0,00	0,00	0,00	30,00	0,00	98,00	0,00	20,00

У таблиці 4.10 представлений узагальнений результат виявлення і класифікації кожного з 22 типів атак десятьма НІД.

Аналізуючи результати проведених експериментів, можна зробити висновок, що запропонована ІІТ виявлення і класифікації атак на ІТМ здатна ефективно виявляти різноманітні мережеві вторгнення. Так, десять детекторів достатньо ефективно виявляють 22 типи мережевих атак. Це

відбувається завдяки узагальнюючій властивості нейронної мережі, яка покладена в основу детектора. При навчанні нейронна мережа знаходить приховані взаємозв'язки в даних і в майбутньому при аналізі невідомого образу коректно його класифікує.

Таблиця 4.10 – Узагальнена достовірність виявлення і класифікації мережевих атак

d_back	d_land	d_neptune	d_pod	d_smurf	d_teardrop	p_ipsweep	p_nmap
99,05	100	100	99,89	100	100	99,08	100
p_prtsweep	p_satan	r_ftpwrite	r_gpasswd	r_imap	r_multihop	r_phf	r_spy
99,90	96,04	100	94,34	99,10	97,60	98,70	100
r_wclient	r_wmaster	u_overflw	u_ldmodul	u_perl	u_rootkit		
90,00	90,00	99,00	100	97,00	98,00		

При підвищенні кількості імунних детекторів, достовірність виявлення і класифікації збільшуватиметься, проте неістотно, через те, що імунні детектори частково «перекривають» самі себе, тобто виявляють і класифікують типи атак, навчання на яких не проводилося. Так, доцільним є дослідження з кількістю імунних детекторів, рівною кількості типів мережевих атак. У таблиці 4.11 представлені результати аналізу мережевого трафіку двадцятьма двома НІД.

Таблиця 4.11 – Достовірність виявлення і класифікації мережевих атак двадцятьма двома НІД

d_back	d_land	d_neptune	d_pod	d_smurf	d_teardrop	p_ipsweep	p_nmap
100	100	100	100	100	100	99,90	100
p_prtsweep	p_satan	r_ftpwrite	r_gpasswd	r_imap	r_multihop	r_phf	r_spy
99,99	99,99	100	99,85	99,89	99,30	100	100
r_wclient	r_wmaster	u_overflw	u_ldmodul	u_perl	u_rootkit		
99,50	99,60	99,20	100	100	98,60		

Як видно з таблиці, достовірність виявлення і класифікації мережевих атак збільшилася, проте неістотно. Середнє значення підвищення достовірності складає 0,3%. Якщо ж збільшити кількість детекторів в 2 рази, тобто їх число становитиме 44, то приріст достовірності виявлення і

класифікації мережевих атак складе лише 0,09%.

На рисунках 4.18-4.19 педставлено порівняльний аналіз розробленої технології виявлення і класифікації атак на ІТМ з існуючими відомими технологіями.

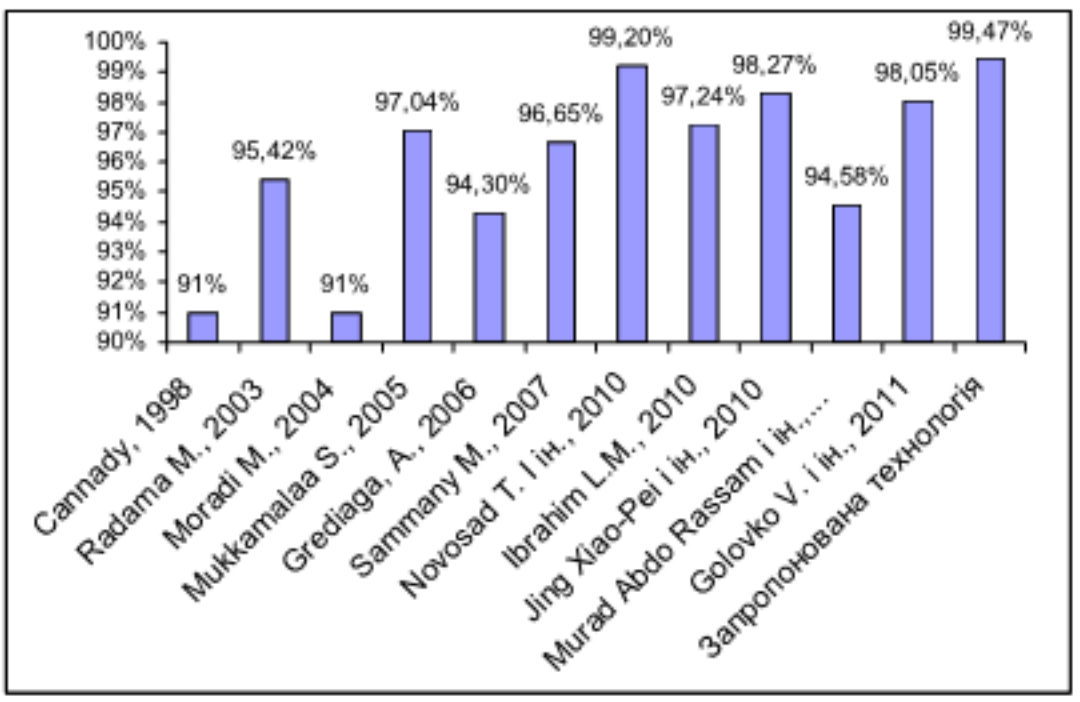


Рисунок 4.18 – Порівняльний аналіз розробленої технології виявлення і класифікації атак на ІТМ з існуючими відомими технологіями (достовірність виявлення і класифікації атак)

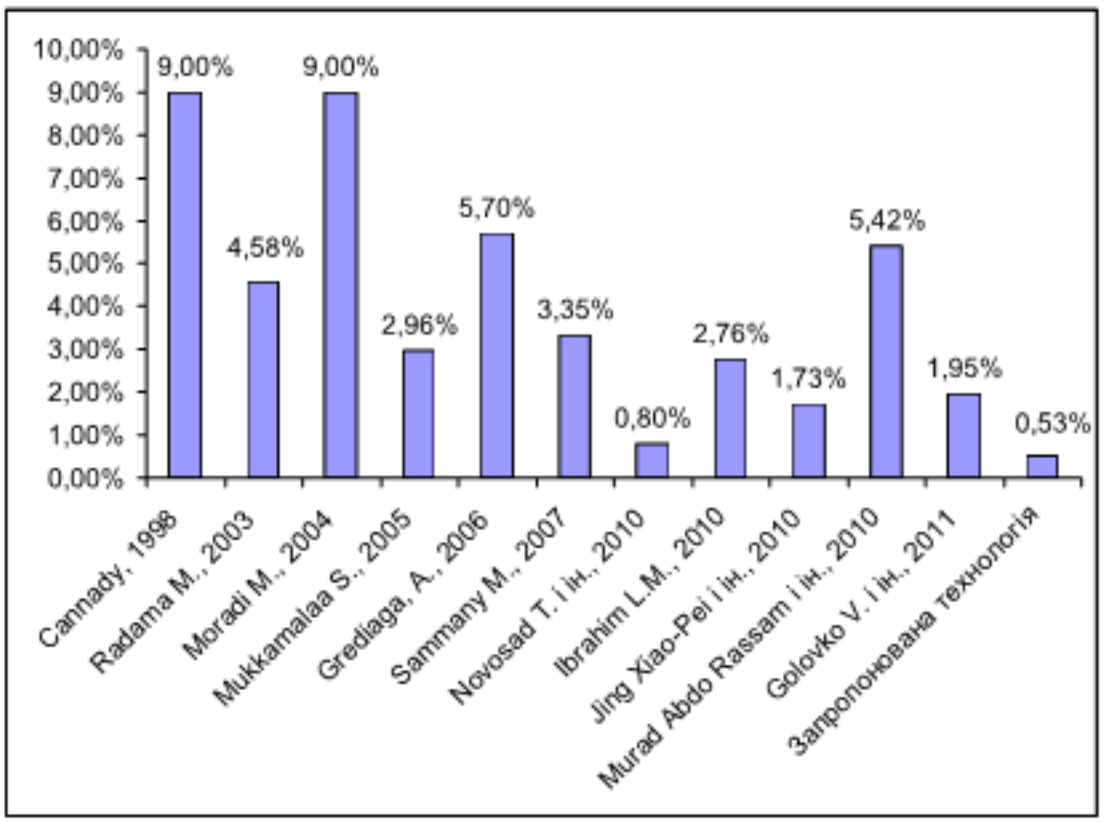


Рисунок 4.19 – Порівняльний аналіз розробленої технології виявлення і класифікації атак на ІТМ з існуючими відомими технологіями (помилка другого роду – ймовірність пропуску атак)

Статистична оцінка достовірності запропонованої технології підтвердила здатність НІД коректно виявляти і класифікувати мережеві атаки при низькому рівні виникнення помилок другого роду. Результати проведених експериментів показали, що НІД виявляють і класифікують різнотипні мережеві атаки з узагальненою ймовірністю 99,47%. Окрім цього, різні детектори здатні з ймовірністю до 100% виявляти і класифікувати деякі інші типи атак, навчання на яких не проводилося. При цьому, рівень помилок першого роду (ймовірність пропуску атаки) становить 0,53%, що в 1,5–17 разів менше в порівнянні з відомими підходами.

У даному підрозділі проведена статистична оцінка достовірності інтелектуальної інформаційної технології, яка показала підвищення достовірності виявлення і класифікації як відомих, так і невідомих атак на ІТМ.

Висновки до розділу 4

1. Розроблено і практично реалізовано на основі запропонованих методів інтелектуальну інформаційну технологію, яка характеризується генеруванням множини нейромережевих детекторів для кожного класу мережевої атаки, що дало можливість виявляти нові атаки та підвищити достовірність виявлення і класифікації атак на ІТМ.

2. Інтелектуальну інформаційну технологію практично реалізовано в рамках нейромережевої імунної системи виявлення і класифікації атак на ІТМ. На основі модульного принципу розроблено програмні засоби і алгоритми функціонування системи. Для програмної реалізації використано мову програмування С++, а засобом реалізації вибрано середовище програмування С++ Builder 7.0.

3. Результати експериментальних досліджень з використанням розробленого програмного забезпечення підтверджують вірність наукових положень запропонованої інформаційної технології, оскільки її

впровадження дало можливість підвищити достовірність виявлення і класифікації атак на ІТМ на 0,27-8,5% у порівнянні з відомими рішеннями. Окрім цього, різні детектори здатні з ймовірністю до 100% виявляти і класифікувати нові атаки. При цьому, рівень помилок першого роду (ймовірність пропуску атаки) складає 0,53%, що в 1,5–17 разів менше в порівнянні з відомими підходами. Рівень помилок другого роду (ймовірність помилкових спрацювань) – 0,6%.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну задачу розробки і дослідження інтелектуальної інформаційної технології на базі теорії штучних нейронних мереж і штучних імунних систем з метою підвищення достовірності виявлення і класифікації атак на ІТМ.

При цьому отримані наступні основні результати:

1. На основі огляду відомих підходів і порівняльного аналізу існуючих інформаційних технологій виявлення і класифікації атак на ІТМ обгрунтовано доцільність створення нової інформаційної технології на базі штучних нейронних мереж і штучних імунних систем з використанням, в якості тестових наборів, часто вживаної бази даних KDD. Показано, що одним з основних критеріїв оцінки ефективності інформаційної технології виявлення і класифікації мережевих атак є достовірність результатів її функціонування, тому в якості засобу оцінки достовірності результатів запропоновано вибрати ROC-аналіз. На цій основі обґрунтовано шляхи вдосконалення методів виявлення і класифікації атак на ІТМ та сформульовано задачі дослідження.

2. Запропоновано узагальнену функціональну модель прийняття рішень при виявленні і класифікації атак на ІТМ у розрізі наступних етапів: отримання параметрів мережевих з'єднань, обробка інформації, формування рішення нейромережевим аналізатором, що дозволило підвищити достовірність виявлення і класифікації мережевих атак.

3. На базі результатів теоретичних та експериментальних досліджень можливих нейронних мереж для виявлення і класифікації атак на ІТМ, в якості основи нейромережевого детектора, вибрано нейронну мережу векторного квантування LVQ з нейронними елементами Кохонена в прихованому шарі, що дозволило суттєво зменшити розмірність навчальної вибірки (в кращому випадку до 20 векторів) і, відповідно, час навчання.

4. Розроблено та досліджено метод побудови нейромережевих

детекторів атак, який базується на нейронній мережі векторного квантування LVQ, де 80% нейронних елементів Кохонена в прихованому шарі відповідають типу атаки, а решта – нормальному з'єднанню. Запропонований метод характеризується малим об'ємом навчальної вибірки і структурою нейронної мережі, в якій співвідношення атак і нормальних з'єднань дорівнює 4:1, що дозволило окремо здійснити кластеризацію атак і нормальних з'єднань в прихованому шарі в порівнянні з іншими підходами, а в результаті підвищити достовірність виявлення і класифікації атак на ІТМ. В рамках цього методу розроблено алгоритм навчання нейромережевого детектора, який враховує особливості його структури.

5. Розроблено та досліджено метод побудови сукупного класифікатора для ієрархічної класифікації атак на ІТМ на основі багатоканальних нейромережевих детекторів, що дало можливість зменшити розмірність аналізованої інформації в 3,4 рази при втраті відносної інформативності 0,8% за рахунок стиснення вхідної інформації (для отримання найбільш інформативних ознак), а також класифікувати мережеві атаки за рахунок об'єднання навчених на певний тип атаки нейромережевих детекторів. Запропонований метод дозволив усунути конфлікти в роботі нейромережевих детекторів.

6. Розроблено та досліджено комбінований метод, який базується на інтеграції нейромережевих детекторів в штучну імунну систему, що дозволило нейромережевим імунним детекторам адаптуватися до невідомих атак на ІТМ за рахунок здійснення операцій клонування і мутації, а також підвищило достовірність виявлення і класифікації невідомих атак в 2-3 рази.

7. На основі запропонованих методів розроблено інтелектуальну інформаційну технологію, яка практично реалізована в рамках нейромережевої імунної системи виявлення і класифікації як відомих, так і невідомих атак на ІТМ, характеризується генеруванням множини нейромережевих детекторів для кожного типу мережевої атаки. Розроблені алгоритми функціонування системи.

8. Результати експериментальних досліджень підтверджують вірність наукових положень дисертаційної роботи, а впровадження запропонованої інформаційної технології дало можливість підвищити достовірність виявлення і класифікації атак на ІТМ на 0,27-8,5% у порівнянні з відомими рішеннями. При цьому рівень помилок першого роду (ймовірність пропуску атаки) склав 0,53%, тобто рівень даних помилок знижено не менше як в 1,5 рази, а рівень помилок другого роду (ймовірність помилкових спрацювань) – 0,6%. Що торкається достовірності виявлення і класифікації невідомих атак, то вона, в окремих випадках, може досягати 100%.

9. Теоретичні та практичні результати, отримані в дисертаційній роботі, використані приватним підприємством «МагнетікВан» (м. Тернопіль), товариством з обмеженою відповідальністю «СофтІнвест» (м. Брест, Республіка Білорусь), Науково-дослідним інститутом інтелектуальних комп'ютерних систем в рамках науково-дослідної роботи, а також в рамках двостороннього договору про партнерство, співпрацю і науковий обмін між Тернопільським національним економічним університетом і Брестським державним технічним університетом і в навчальному процесі Тернопільського національного економічного університету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технология «гибридной» защиты в продуктах Лаборатории Касперского [Електронний ресурс] – Режим доступу : http://www.anti-malware.ru.
2. Kaspersky Security Bulletin. Основная статистика за 2011 год [Електронний ресурс] – Режим доступу : http://www.securelist.com/ru/analysis/208050741/Kaspersky_Security_Bulletin_Osnovnaya_statistika_za_2011_god#11.
3. Golovko V. Neural Networks approaches for Intrusion Detection and Recognition / V. Golovko, L. Vaitsekhovich // Computing. – 2006. – Vol. 5, N.3. – P. 118–125.
4. Dimensionality Reduction and Attack Recognition using Neural Network Approaches / V. Golovko, L. Vaitsekhovich, P. Kochurko, V. Rubanau // Proceedings of the International Joint Conference on Neural Networks (IJCNN 2007), Orlando, FL, USA – Orlando, 2007. – P. 2734-2739.
5. Головко В.А. Нейронная сеть как элемент системы обнаружения вторжений / В.А. Головко, Л.Ю. Войцехович // X Всероссийская научно-техническая конференция «Нейроинформатика-2008»: сборник научных трудов в 2-х частях. – Москва: Московский инженерно-физический институт, 2008. – Ч.2.– С.86–93.
6. Golovko V. Boosting Algoritms for Ensemblers of Neural Network Classifiers in Intrusion Detection Domain / V. Golovko, L. Vaitsekhovich // Proceedings of the International Conference on Neural Networks and Artificial Intelligence (ICNNAI-2008) – Minsk, Belarus, 2008. – P. 70-74.
7. Vaitsekhovich L. Multiagent Intrusion Detection Based on Neural Networks Detectors and Artificial Immune System / L.Vaitsekhovich, V. Golovko, U. Rubanau // Proceedings of the 10th International Conference on Pattern Recognition and Information Processing (PRIP-2009) – Minsk, Belarus, 2008. – P. 70–74.
8. Neural Network and Artificial Immune Systems for Malware and Network

Intrusion Detection / V. Golovko, S. Bezobrazov, P. Kachurka, L. Vaitsekhovich // Studies in computational intelligence. – Springer Berlin/Heidelberg, 2010. – Vol. 263: Advances in machine learning II. – P. 485–513.

9. Кочурко П.А. Построение нейросетевой системы обнаружения и распознавания атак / Кочурко П.А., Головко В.А. // Вестник Брестского государственного технического университета: (Серия: физика, математика и информатика). – 2010. – №5. – С. 7-13.
10. Войцехович Л.Ю. Построение системы обнаружения атак с использованием графа взаимодействия агентов / Л.Ю. Войцехович, В.А. Головко, Курош Мадани // Вестник Брестского государственного технического университета: (Серия: физика, математика и информатика). – 2010. – №5. – С. 17–21.
11. Головко В.А. Проектирование интеллектуальных систем обнаружения аномалий // В.А. Головко, С.В.Безобразов // Международная научно-техническая конференция «Open Semantic Technologies for Intelligent Systems (OSTIS-2011) – Minsk, Belarus. 2011. – P. 70–74.
12. Войцехович Л.Ю. Применение мультиагентной системы с нейросетевым классификатором для выявления атак в трафике TCP/IP / Л.Ю. Войцехович, В.А. Головко, Курош Мадани // Научная сессия МИФИ – 2011. Сборник научных трудов. – 2011. – Ч.1. – С. 190–202.
13. Asynchronous alert correlation in multi-agent intrusion detection systems / V. Gorodetsky, O. Karsaev, V. Samoilov, A. Ulanov // Springer-Verlag, Berlin, Heidelberg, 2005. – Vol. 3685. – P. 366–379.
14. Gorodetsky V. Multi-agent Technologies for Computer Network Security: Attack Simulation, Intrusion Detection and Intrusion Detection Learning / V. Gorodetsky, I. Kotenko, O. Karsaev // International Journal of Computer Systems Science and Engineering. 2003. – Vol.18, №4. – P.191–200.
15. Многоагентная система обучения обнаружению атак / В.И. Городецкий, О.В. Карсаев, И.В. Котенко, [и др.] // Материалы Межрегиональной

конференции «Информационная безопасность регионов России» (ИБРР-2003). Часть 1. СПб, 2003. – С.118.

16. Городецкий В.И. Программный прототип многоагентной системы обнаружения вторжений в компьютерные сети / Городецкий В.И., Карсаев О.В., Котенко И.В. // Труды Международного конгресса «Искусственный интеллект в XXI веке». Том 1. М.: Физматлит, 2001. – С. 280–293.
17. Городецкий В.И. Многоагентная система защиты информации в компьютерных сетях: механизмы обучения и формирования решений для обнаружения вторжений / Городецкий В.И., Котенко И.В., Карсаев О.В. // Труды Межрегиональной конференции «Информационная безопасность регионов России» (ИБРР-99). СПб., 2000. – С.97–104.
18. Чечулин А.А. Обнаружение и противодействие сетевым атакам на основе комбинированных механизмов анализа трафика / А.А.Чечулин, И.В.Котенко // Материалы XVIII Общероссийской научно-технической конференции «Методы и технические средства обеспечения безопасности информации». СПб.: Издательство Политехнического университета. – 2009. – С.69.
19. Котенко И.В. Многоагентное моделирование механизмов защиты от распределенных компьютерных атак / И.В.Котенко, А.В. Уланов // Информационные технологии. – 2009. – № 2. – С.38–44.
20. Котенко И.В. Интеллектуальные механизмы управления кибербезопасностью / И.В. Котенко // Управление рисками и безопасностью. Труды Института системного анализа Российской академии наук (ИСА РАН). М.:, 2009. – Т.41.– С.74–103.
21. Котенко И.В. Исследование механизмов защиты от атак DDOS: имитация противоборства интеллектуальных агентов в сети Интернет / И.В. Котенко, А.В. Уланов // Международная конференция «РусКрипто'2008» [Електронний ресурс] – Режим доступу :

http://www.ruscrypto.ru.

22. Shyrochin V.P. Users Behavior Model in Tasks of Computer Systems Security Analysis / V.P Shyrochin., V.E. Mukhin, Hu Zhengbing // Computing. – 2003. – № 2. – P. 151–156.
23. Широчин В.П. Методы и средства анализа безопасности корпоративных сетей / В.П. Широчин, В.Е. Мухин В.Е., Ху Чженбин // Труды 4-ой международной научно-практической конференции «Современные информационные и электронные технологи», Одесса (Украина), 2003. – С.64.
24. Широчин В.П. Два подхода к обнаружению попыток НСД в компьютерных системах / В.П. Широчин, Ху Чженбин // Труды 5-ой международной научно-практической конференции «Современные информационные и электронные технологи», Одесса (Украина), 2004. – С.81.
25. Широчин В.П. Обнаружение аномалий на основе неконтролируемой кластеризации / В.П. Широчин, Ху Чженбин, Д.Г. Гундарцев // Труды 6-ой международной научно-практической конференции «Современные информационные и электронные технологи», Одесса (Украина), 2005. – С.116.
26. Широчин В.П. Средства сопровождения адаптивных комплексных систем защиты информации / В.П. Широчин, Ху Чженбин // Сборник трудов IV международной научно-практической конференции «Проблемы внедрения информационных технологий в экономике и би знесе», Ирпень (Украина), 2003. – С. 661–664.
27. Широчин В.П. Механизмы и средства мониторинга безопасности информационных систем / В.П. Широчин, В.Е. Мухін, Ху Чженбин // Сборник трудов IV международной научно-практической конференции «Проблемы внедрения информационных технологий в экономике и би знесе», Ирпень (Украина), 2004. – С. 519–522.
28. Shyrochin V.P. Users Behavior Model in Tasks of Computer Systems

Security Analysis / V.P. Shyrochin, V.E. Mukhin, Hu Zhengbing // Proceedings IEEE Second International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2003), Lviv (Ukraine), 2003. – P. 463–466.

29. Shyrochin V.P. Data Mining Approaches for Signatures Search In Network Intrusion Detection / V.P. Shyrochin, Hu Zhengbing // Proceedings IEEE Second International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2005), Sofia (Bulgaria), 2005. – P. 363–367.
30. Hu Zhengbing. A Reasonable Complement for the Firewall - IDSs / Hu Zhengbing, V.P. Shyrochin, Su Jun // The 10-th National Youth Communication Conference of China, 2005. – P. 258–261.
31. Cannady J. Artificial neural networks for misuse detection / J. Cannady // Proceedings of the 21st national information systems security conference. – Arlington, VA, USA, 1998. – October, 5–8. – P. 368–381.
32. Dasgupta D. A Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection / D. Dasgupta, F. Nino // In the Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC). – Nashville, 2000. – October, 8–11. – P. 125–130.
33. De Castro L. N. Artificial Immune Systems as a Novel Soft Computing Paradigm / L. N. De Castro, J. Timmis // Soft Computing Journal. – 2003. – Vol. 7, Issue 7. – P. 268–284.
34. De Castro L. N. Artificial Immune Systems: Part I – Basic Theory and Applications / L. N. De Castro, F. J. Von Zuben // Technical Report – RT DCA 01/99, FEEC/UNICAMP. – Brazil, 1999. – 95 p.
35. De Castro L. N. Artificial Immune Systems: Part II – A Survey of Applications / L. N. De Castro, F. J. Von Zuben // Technical Report – RT DCA 02/00, FEEC/UNICAMP. – Brazil, 2000. – 64 p.
36. Application of neural networks in network control and information security / A. Grediaga, F. Ibarra, F. García [et al.] // Springer: Lecture Notes in

Computer Science. – 2006. – Vol. 3973. – P. 208–213.

37. Ibrahim L.M. Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN) / L.M. Ibrahim // Journal of Engineering Science and Technology. – 2010. – Vol. 5, № 4. – P. 457–471.
38. Xiao-Pei Jing. A new Immunity Intrusion Detection Model Based on Genetic Algorithm and Vaccine Mechanism / Jing Xiao-Pei, Wang Hou-Xiang // Computer Network and Information Security. – 2010. – Vol. 2. – P. 33–39.
39. Moradi M. A neural network based system for intrusion detection and classification of attacks / M. Moradi, M. Zulkernine // IEEE International Conference on Advances in Intelligent Systems – Theory and Applications. – Luxembourg-Kirchberg, 2004. – [Електронний ресурс]. – Режим доступу: http://research.cs.queensu.ca/~moradi/148-04-MM-MZ.pdf.
40. Mukkamalaa S. Intrusion detection using an ensemble of intelligent paradigms / S. Mukkamalaa, A.H. Sung, A. Abraham // Journal of Network and Computer Applications. – 2005. – Vol. 28, № 2. – P. 167–182.
41. Murad Abdo Rassam. Intrusion Detection System Using Unsupervised Immune Network Clustering with Reduced Features / Murad Abdo Rassam, Mohd. Aizaini Maarof, Anazida Zainal // Int. J. Advance. Soft Comput. Appl. – 2010. – Vol. 2, № 3. – P. 244–263.
42. Fast intrusion detection system based on flexible neural tree / T. Novosad, J. Platos, V. Snasel, A. Ajith // Proceedings of the Sixth international conference on information assurance and security (IAS). – USA, 2010. – P. 142–147.
43. Ramadas M. Detecting anomalous network traffic with self-organizing maps / M. Ramadas, S. Ostermann, B. Tjaden // Springer: Lecture Notes in Computer Science. – 2003. – Vol. 2820. – P. 36–54.
44. Artificial neural networks architecture for intrusion detection systems and classification of attacks / M. Sammany, M. Sharawi, M. El-Beltagy, I. Saroit // Proceedings of the 5th international conference INFO2007. – [Електронний ресурс]. – Режим доступу: http://www.academia.edu/1419323/

Artificial_Neural_Networks_Architecture_For_Intrusion_Detection_Systems _and_Classification_of_Attacks.

45. Sridevi R. Analysis of Human Immune System Inspired Intrusion Detection System / R. Sridevi, Dr. Rajan Chattamvelli, Dr.E.Kannan // International Journal of Computer Science and Information Technologies. – 2011. – Vol. 2, № 5. – P. 2335–2339.
46. Комар М.П. Підхід до ідентифікації комп'ютерних атак засобами інтелектуальних інформаційних технологій / М.П. Комар // Збірник тез другої міжнародної науково-практичної конференції «Комп'ютерні системи та мережеві технології» (CSNT- 2009). – Київ, 2009. – С.53.
47. Комар М.П. Использование искусственных иммунных систем и нейронных сетей для обнаружения компьютерных атак / М.П. Комар // Сборник VI Республиканской научной конференции молодых ученых и студентов «Современные проблемы математики и вычислительной техники». – Брест (Республика Беларусь), 2009. – Т.1. – С. 16–18.
48. Комар М.П. Методы искусственных нейронных сетей для обнаружения сетевых вторжений / М.П. Комар // Збірник тез сьомої Міжнародної науково-технічної конференції «Інтернет – Освіта – Наука» (ІОН-2010). – Вінниця (Україна), 2010. – С. 410–413.
49. Комар М.П. Нейромережевий метод ідентифікації комп'ютерних атак / М.П. Комар // Оптико-електронні інформаційно-енергетичні технології. – 2010. – №2. – С. 105–109.
50. Комар М.П. Система анализа сетевого трафика для обнаружения компьютерных атак / М.П. Комар // Вестник Брестского государственного технического университета: (Серия: физика, математика и информатика). – 2010. – №5. – С. 14–16.
51. Комар М.П. Алгоритми штучних нейронних мереж для виявлення мережевих вторгнень / М.П. Комар // Збірник тез Міжнародної науково-технічної конференції молодих вчених та студентів „Актуальні задачі сучасних технологій". – Тернопіль (Україна), 2010. – С. 89.

52. Комар М.П. Використання методу головних компонент для вирішення задачі виявлення комп'ютерних атак / М.П. Комар // Збірник тез третьої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації». – Вінниця (Україна), 2011. – С. 131–132.
53. Комар М.П. Інформаційна модель процесу виявлення комп'ютерних атак на основі нейромережевих класифікаторів / М.П. Комар // Збірник тез першої Міжнародної науково-технічної конференції «Computational Intelligence» (CI-2011). – Черкаси (Україна), 2011. – С. 179–180.
54. Комар М.П. Методы искусственных иммунных систем и нейронных сетей для обнаружения компьютерных атак / М.П. Комар // Інформаційна безпека. – 2011. – №1(5). – С. 154–160.
55. Комар М.П. Інтелектуальна система виявлення мережевих атак на інформаційні ресурси на основі методу головних компонент / М.П. Комар // Системи обробки інформації. – 2011. – №8(98). – С. 203–207.
56. Комар М.П. Інтелектуальна інформаційна технологія виявлення мережевих атак у системі реального часу / М.П. Комар // Радіоелектронні і комп'ютерні системи. – 2011. – № 4(52). – С. 92-98.
57. Комар М.П. Метод построения совокупного классификатора трафика информационно-телекоммуникационных сетей для иерархической классификации компьютерных атак / М.П. Комар // Системи обробки інформації. – 2012. – №3(101). – С. 134–138.
58. Golovko V. Principles of neural network artificial immune system design to detect attacks on computers / V. Golovko, M. Komar, A. Sachenko // Proceedings of the Xth International Conference «Modern Problems of Radio Engineering, Telecommunications and Computer Science» (TCSET'2010). – Lviv-Slavsko (Ukraine), 2010. – P. 237.
59. Комар М.П. Інтелектуалізована інформаційна технологія виявлення комп'ютерних атак / М.П. Комар, Д.І. Боднар, А.О. Саченко //

Вимірювальна та обчислювальна техніка в технологічних процесах. – 2010. – №2. – С. 133–137.

60. Komar M. Intelligent system for detection of networking intrusion / M. Komar, V. Golovko, A. Sachenko, S. Bezobrazov // Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS-2011). – Prague (Czech Republic), 2011. – V1. – P. 374-377.
61. Golovko V. Evolution of Immune Detectors in Intelligent Security System for Malware Detection / V. Golovko, S. Bezobrazov, V. Melianchuk, M. Komar // Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS-2011). – Prague (Czech Republic), 2011. – V1. – P. 722–726.
62. Комар М.П. Нейросетевой подход к обнаружению компьютерных атак на информационные ресурсы / М.П. Комар, И.О. Палий, Р.П. Шевчук, Т.Б. Федысив // Інформатика та математичні методи в моделюванні. – 2011. – Т. 1, №2. – С. 156–163.
63. Комар М.П. Інтеграція нейромереж та штучних імунних систем для виявлення комп'ютерних атак / М.П. Комар, А.О. Саченко, В.А. Головко, Т.Г. Фортуна // Матеріали І-ої Міжнародної науково-технічної конференції «Захист інформації і безпека інформаційних систем». – Львів (Україна), 2012. – С. 108–109.
64. Komar M. Method of Aggregate Classifier Construction for Hierarchical Classification of Computer Attacks / M. Komar, V. Golovko, O. Lyashenko, A. Sachenko // Proceedings of the XX Ukrainian-Polish conference «CAD in Machinery Design. Implementation and Educational Issues» (CADMD 2012. – Lviv, Ukraine, 2012. – P. 80–82.
65. Пат. №74822 Україна, МПК(2012) H04W 12/08, G06F 21/00, G06F 12/14. Спосіб виявлення комп'ютерних атак нейромережевою штучною імунною системою / Комар М.П., Саченко А.О., Головко В.А., Безобразов С.В.; заявник і патентовласник Тернопільський національний

економічний університет. – № u201205349 ; заявл. 28.04.12 ; опубл. 12.11.12, Бюл. № 21.

66. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы. Учебник для ВУЗов. 4-ое издание / В.Г. Олифер, Н.А. Олифер. – СПб: Питер, 2010. – 943 с.
67. Закон України «Про телекомунікації» від 18 листопада 2003 р. // Відомості Верховної Ради. – 2003. – №56. – с.447.
68. Системи технічного захисту інформації. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу : НД ТЗІ 1.1-003-99 . – [Чинний від 01.07.99]. – К.: – ДСТСЗІ СБ України, 1999. – 24 с. – (Нормативний документ).
69. Захист інформації. Технічний захист інформації. Терміни та визначення : ДСТУ 3396.2-97. – [Чинний від 01.01.98]. – К.: Держстандарт України, 1998. – 12 с. – (Державний стандарт України).
70. Компьютерные атаки и технологии их обнаружения [Електронний ресурс] – Режим доступу : http://web-protect.net/attack.htm.
71. Что такое сетевая атака? [Електронний ресурс] – Режим доступу : http://bestprogi.ru/wopros8.html.
72. Удалённые сетевые атаки [Електронний ресурс] – Режим доступу : http://ru.wikipedia.org/wiki/Удалённые_сетевые_атаки.
73. Автоматизовані системи. Терміни та визначення : ДСТУ 2226-93. – [Чинний від 01.07.94]. – К.: Держстандарт України, 1998. – 12 с. – (Державний стандарт України).
74. Системи оброблення інформації. Інтелектуальні інформаційні технології. Терміни та визначення : ДСТУ 2481-94. – [Чинний від 01.01.95]. – К.: Держстандарт України, 1998. – 12 с. – (Державний стандарт України).
75. Атака через Интернет / И.Д. Медведовский, П.В.Семьянов, Д. Г. Леонов, А. В. Лукацкий. – М. : Солон – Р, 2005. – 368 с.
76. Черней Г.А. Безопасность автоматизированных информационных систем / Г.А. Черней, С.А. Охрименко, Ф.С. Ляху. – Кишинев : Руханда, 1996. –

186 с.

77. Гайкович В. Безопасность электронных банковских систем / В. Гайкович, А. Першин. – М.: Изд-во компания «Единая Європа», 1994. – 363 с.
78. Системы обнаружения атак на сетевом уровне. [Електронний ресурс]. – Режим доступу: http://www.citforum.ru/intemet/securities/faq ids.shtml. – Назва з екрану.
79. Лукацкий А.В. Мир атак многообразен. [Електронний ресурс]. – Режим доступу: http://www.infosec.ru//press/pub_luka.html.
80. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – СПб.: БХВ-Петербург, 2003. – 596 с.
81. KDD Cup 1999 Data [Електронний ресурс]. – Режим доступу: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
82. Межсетевой экран // Википедия [Електронний ресурс]. – Режим доступу: http://ru.wikipedia.org/wiki/Межсетевой экран.
83. Касперский К. Техника сетевых атак / К. Касперский. – М.: Солон – Р, 2001. – 396 с.
84. Брэгг Р. Безопасность сетей. Полное руководство / Р. Брэгг, М. Родс-Оусли, К. Страссберг. – М.: Эком, 2006. – 912 с.
85. Система обнаружения вторжений // Википедия [Електронний ресурс]. – 2005. – Режим доступу: http://ru.wikipedia.org/wiki/Система обнаружения вторжений.
86. How an intrusion detection system operates // Intrusion detection system [Електронний ресурс]. – 2011. – Режим доступу: http://www.intrusiondetectionsystem.org.
87. Бил Д. Snort 2.1. Обнаружение вторжений / Д. Бил, Э. Бейкер, Б. Казуэлл. – М.: Бином-Пресс, 2011. – 656 с.
88. Bray R. OSSEC host-based intrusion detection guide / R. Bray. – Elsevier, 2010. – 416 с.
89. Prelude hybrid IDS // Wikipedia [Електронний ресурс]. – 2005. – Режим

доступу: http://en.wikipedia.org/wiki/Prelude_hybrid_IDS.

90. Обзор методов обнаружения атак на информационную систему [Електронний ресурс] – Режим доступу : http://inf-bez.ru/?p=668.
91. Lia L.B. Detecting network intrusions using signal processing with query-based sampling filter / L.B. Lia, R.I. Chang, J.S. Kouh // EURASIP Journal on Advances in Signal Processing. – 2009. – Article ID 735283. – P. 1–8.
92. Morteza A. A practical solution to real-time network-based intrusion detection using unsupervised neural networks / A. Morteza, R. Jalili, R.S. Hamid // Computers & Security. – 2006. – Vol. 25, № 6. – P. 459 – 468.
93. Chen W.H. Application of SVM and ANN for intrusion detection / W.H. Chen, S.H. Hsu, H.P. Shen // Computers & Operations Research. – 2005. – Vol. 32, № 10. – P. 2617–2634.
94. Alfantookh A.A. DoS attacks intelligent detection using neural networks / A.A. Alfantookh // Comp. & Info. Sci. –2006. – Vol. 18. –P. 27–45.
95. Intrusion detection method using neural networks based on the reduction of characteristics / I. Lorenzo-Fonseca, F. Maciá-Pérez, F. Mora-Gimeno [et al.] // Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence. – Springer-Verlag Berlin, Heidelberg, 2009. – P. 1296–1303.
96. InSeon Y. An intelligent firewall to detect novel attacks? An integrated approach based on anomaly detection against virus attacks / Y. InSeon, U. Ulrich // SOFSEM 2002 Student Research Forum. – 2002. – P. 59–64. – [Електронний ресурс]. – Режим доступу: http://www2.fiit.stuba.sk/~bielik/sofsem2002srf/clanky/09Yoo.pdf.
97. A mutated intrusion detection system using principal component analysis and time delay neural network / B.D. Kang, J.W. Lee. J.H. Kim [et al.] // Springer: Lecture Notes in Computer Science. –2006. – Vol. 3973. – P. 246–254.
98. Siddiqui M.A. High performance data mining techniques for intrusion detection: Master's Thesis / M.A. Siddiqui. – University of Engineering & Technology, School of Computer Science, College of Engineering &

Computer Science at the University of Central Florida. – [Електронний ресурс]. – Режим доступу: http://etd.fcla.edu/CF/CFE0000056/Siddiqui_Muazzam_A_200405_MS.pdf.

99. Zhang C. Comparison of BPL and RBF Network in intrusion detection system / C. Zhang, J. Jiang, M. Kamel // Springer: Lecture Notes in Computer Science. – 2004. – Vol. 2639. – P. 460–470.
100. Cannady J. Applying CMAC-based online learning to intrusion detection / J. Cannady // Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000). – 2005. – P. 405–410.
101. Debar H. A neural network component for an intrusion detection system / H. Debar, M. Becker, D. Siboni // IEEE Computer Society Symposium on Research in Security and Privacy. – 1992. – P. 240–250.
102. Network-based anomaly detection using an Elman network / E. Cheng, H. Jin, Z. Han, J. Sun // Networking and Mobile Computing. – 2006. – Vol. 3619. – P. 471–480.
103. Fox K.L. A neural network approach towards intrusion detection / K.L. Fox, R.R. Henning, J.H. Reed // Proceedings of the 13th National Computer Security Conference. – [Електронний ресурс]. – Режим доступу: http://csrc.nist.gov/nissc/2000/proceedings/papers/045.pdf.
104. Rhodes B.C. Multiple self-organizing maps for intrusion detection / B.C. Rhodes, J.A. Mahaffey, J.D. Cannady // Proceedings of the 23rd National Information Systems Security Conference. – Baltimore, MA, USA, 2000. – October, 16–19. – P. 16–19.
105. Höglund A.J. A computer host-based user anomaly detection system using the self-organizing map / A.J. Höglund, K. Hätönen, A.S. Sorvari / Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00). – 2000. – Vol. 5. – P. 411–416.
106. Lichodzijewski P. Host-based intrusion detection using self-organizing maps / P. Lichodzijewski, A.N. Zincir-Heywood, M.I. Heywood // Proceedings of the International Joint Conference on Neural Networks (IJCNN'02). – 2002. – P.

1714–1719.

107. Sarasamma S.T. Hierarchical Kohonen net for anomaly detection in network security / S.T. Sarasamma, Q.A. Zhu, J. Huff // IEEE Transaction on Systems, Man and Cybernetics. – 2005. – Vol. 35, № 2. – P. 302–312.
108. Jirapummin C. Hybrid neural networks for intrusion detection system / C. Jirapummin, N. Wattanapongsakorn, P. Kanthamanon // Proceedings of the 2002 International Technical Conference On Circuits/Systems, Computers and Communications. – 2002. – P. 228–237.
109. Horeis T. Intrusion detection with neural network – Combination of selforganizing maps and redial basis function networks for human expert integration / T. Horeis // [Електронний ресурс]. – Режим доступу: http://ieee-cis.org/_files/EAC_Research_2003_Report_Horeis.pdf.
110. Chimphlee W. Anomaly-based intrusion detection using fuzzy rough clustering / W. Chimphlee, A.H. Abdullah, M.N.M. Sap // International Conference in Hybrid Information Technology (ICHIT'06). – 2006. – Vol. 1. – P. 329–334.
111. Fuzzy intrusion detection / J.E. Dickerson, J.Juslin, J. Koukousoula, J.A. Dickerson // Proceedings of the 20th International Conference of the North American Fuzzy Information Society (NAFIPS'01) and Joint the 9th IFSA World Congress. – Vancouver, Canada, 2001. – July, 25-28. – Vol. 3. – P. 1506–1510.
112. Diaz-Gomez P.A. Improved off-line intrusion detection using a genetic algorithm / P.A. Diaz-Gomez, D.F. Hougen // Proceedings of the Seventh International Conference on Enterprise Information Systems. –2005. –P. 66–73.
113. Diaz-Gomez P.A. A genetic algorithm approach for doing misuse detection in audit trail files / P.A. Diaz-Gomez, D.F. Hougen // The 15th International Conference on Computing (CIC'06). – 2006. – P. 329–338.
114. Dasgupta D. Artificial Immune Systems and Their Applications / D. Dasgupta – Berlin: Springer-Verlag, Inc., 1999. – 306 p.

115. Ройт А. Основы иммунологии / А. Ройт. – М.: Мир, 1991. – 328 с.
116. Ройт А. Иммунология / А. Ройт, Д. Бростофф, Д. Мейл. – М.: Мир, 2000. – 592 с.
117. Хильгерс А. Иммунная система. Мобилизация внутренних сил / А. Хильгерс, И. Хофман. - СПб.: Питер, 2003. – 192 с.
118. Земсков А.М. Клиническая иммунология / А.М. Земсков, В.М. Земсков, А.В. Караулов ; под ред. А.М. Земскова. – М. : Гэотар медицина, 2008. – 432 с.
119. Койко Р. Иммунология / Р. Койко, Д. Саншайн, Э. Бенджамин. – М.: Академия, 2008. – 365 с.
120. Хайтов Р.М. Иммунология / Р.М. Хайтов. – М.: Гэотар-медиа, 2008. – 256 с.
121. Suzuki J. An Extensible Framework for Simulating Immune Network / J. Suzuki, Y. Yamamoto // Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC). – Nashville, 2000, October, 8–11, – P. 82–96.
122. Dasgupta D. Immunity-Based Intrusion Detection Systems: A General Framework / D. Dasgupta // Proceedings of the 22nd National Information Systems Security Conference (NISSC). – 1999, October, 18–21. – P. 367–381.
123. Dasgupta D. An anomaly detection algorithm inspired by the immune system / D. Dasgupta, S. Forrest // Artificial Immune System and Their Applications. – Springer-Verlag, 1999. – P. 262–277.
124. De Castro L. N. An Evolutionary Immune Network for Data Clustering / L. N. De Castro, F. J. Von Zuben // Proceedings of the IEEE SBRN'00 (Brazilian Symposium on Artificial Neural Networks). – Brazil, 2000. – P. 84–89.
125. De Castro L. N. An Immunological Approach to Initialize Centers of Radial Basis Function Neural Networks / L. N. De Castro, F. J. Von Zuben // Proceedings of the IEEE SBRN'01 (Brazilian Conference on Neural Networks). – Brazil, 2001. – P. 79–84.

126. Dasgupta D. A new Algorithm for Anomaly Detection in Time series Data / D. Dasgupta // International Conference on Knowledge based Computer Systems (KBCS-96). – Bombay, India, 1996, December, 16-18. – P. 174–182.
127. Dasgupta D. Novelty Detection in Time Series Data using Ideas from Immunology / D. Dasgupta, S. Forrest // In ISCA 5th International Conference on Intelligent Systems. – Reno, Nevada, 1996, June, 19–21. – P. 82–87.
128. Dasgupta D. Using Immunological Principles in Anomaly Detection / D. Dasgupta // Proceedings of the Artificial Neural Networks in Engineering (ANNIE'96). – St. Louis, USA, 1996, November, 10–13. – P. 43–51.
129. De Castro L. N. Learning and Optimization Using the Clonal Selection Principle / L. N. De Castro, F. J. Von Zuben // IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems. – 2002. – Vol. 6, № 3. – P. 239–251.
130. De Castro L. N. The Clonal Selection Algorithm with Engineering Applications / L. N. De Castro, F. J. Von Zuben // Proceedings of GECCO'00. – 2000. – P. 36–37.
131. De Castro L. N. Immune and Neural Network Models: Theoretical and Empirical Comparisons / L. N. De Castro, F. J. Von Zuben // International Journal of Computational Intelligence and Applications (IJCIA). – 2001. – Vol. 1, № 3. – P. 239–257.
132. Anchor K.P. The computer defense immune system: current and future research in intrusion detection / K.P. Anchor, P. Williams, G. Gunsch // Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02). – 2002. – Vol. 2. – P. 1027–1032.
133. Dasgupta D. Advances in artificial immune systems / D. Dasgupta // IEEE Computational Intelligence Magazine. – 2006. – Vol. 1. – P. 40–49.
134. De Castro L.N. Artificial immune systems as a novel soft computing paradigm / L.N. de Castro, J.I. Timmis // Soft Computing. – 2003. – Vol. 7. – P. 526–544.

135. Forrest S. Computer immunology / S. Forrest, C. Beauchemin // Immunological reviews. – 2007. – Vol. 1. – P. 176–197.

136. Kim J. Immune system approach to intrusion detection – a review / J. Kim, P. Bentley, U. Aickelin // Natural Computing: An International Journal. – 2007. – Vol. 6. – P. 413–466.

137. Aickelin U. Immune systems approaches to intrusion detection: a review. Artificial Immune Systems / U. Aickelin, J. Greensmith, J. Twycross // Springer: Lecture Notes in Computer Science. – Berlin, 2004. – Vol. 3239. – P. 316–329.

138. Sabhnani M. Application of Machine Learning Algorithms to KDD Intrusion detection dataset within misuse detection context / M. Sabhnani, G. Serpen // Proceedings of the international conference on machine learning: models, technologies and applications. – 2003. – P. 209–215.

139. Kayacik H.G. A hierarchial SOM-based intrusion detection system / H.G. Hayacik, A.N. Zincir-Heywood, M. Heywood // Engeneering applications of artificial intelligence. – 2007. – Vol. 20, № 4. – P. 439–451.

140. Liu G. A hierarchial intrusion detection model based on the PCA neural networks / G. Liu, Z. Yi, S. Yang // Neurocomputing. – 2007. – Vol. 70, № 7–9. – P. 1561–1568.

141. A new approach to intrusion detection using artificial neural networks and fuzzy clastering / G. Wang, J. Hao, J. Ma, L. Huang // Expert systems with applications. – 2010. – Vol. 35, № 3. – P. 1122–1131.

142. Muna M.J. Design network intrusion detection system using hybrid fuzzy-neural network / M.J. Muna, M. Mehrotra // International journal of computer science and security. – 2010. – Vol. 4, № 3. – P. 258–294.

143. Grinstein G. Information Exploration Shootout or «Benchmarks for Information Exploration». / G. Grinstein // Proceedings IEEE Visualization'96. – San Francisco, California, USA, October 1996. – P. 449–450.

144. Grinstein G. Information Exploration Shootout Project and Benchmark Data

Sets: Evaluating how Visualization does in Analyzing Real-World Data Analysis Problems / G. Grinstein // Proceedings IEEE Visualization'97. – Phoenix, Arizona, USA, October 20–21, 1997. – P. 511–513.

145. Costbased modeling for fraud and intrusion detection / S. J. Stolfo, W. Fan, W. Lee [et. al.] // Results from the jam project. – 2000. – Vol. 2. – 1130 p.
146. Evaluating intrusion detection systems / R. P. Lippmann, D. J. Fried, I. Graf [et. al.] // The 1998 darpa off-line intrusion detection evaluation. – 2000. – Vol. 2. – 1012 p.
147. McHugh J. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory / J. McHugh // ACM Transactions on Information and System Security. – 2000. – Vol. 3, № 4. – P. 262–294.
148. MIT Lincoln Labs, 1998 DARPA Intrusion Detection Evaluation [Електронний ресурс]. – Режим доступу: http://www.ll.mit.edu/mission/ communications/ist/corpora/ideval/index.html.
149. Balthrop J. Revisiting LISYS: Parameters and normal behavior / J. Balthrop, S. Forrest, M. R. Glickman // Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02). – Honolulu, HI, USA, 2002. – 12–17 May. – Vol. 2. – P. 1045–1050.
150. Rhodes B. C. Multiple selforganizing maps for intrusion detection / B.C. Rhodes, J.A. Mahaffey, J.D. Cannady / Proceedings of the 23rd National Information Systems Security Conference. – Baltimore, MA, USA, 2000. – 16–19 October. – P. 16–19.
151. Al-Subaie M. The power of temporal pattern processing in anomaly intrusion detection / M. Al-Subaie, M. Zulkernine // IEEE International Conference on Communications (ICC'07). – Glasgow, Scotland, 2007. – 24–28 June. – P. 1391–1398.
152. Using artificial anomalies to detect unknown and known network intrusions / W. Fan, M. Miller, S. Stolfo [et. al.] // Knowledge and Information Systems. – 2004. – Vol. 6, № 5. – P. 507–527.

153. Ghosh A. K. Detecting anomalous and unknown intrusions against programs / A. K. Ghosh, J. Wanken, F. Charron // Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC'98). – Phoenix, AZ, USA, 1998. – 7–11 December. – P. 259–267.

154. González F. Combining negative selection and classification techniques foranomaly detection / F. González, D. Dasgupta, R. Kozma // Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02). – Honolulu, HI, USA, 2002. – 12–17 May. – Vol. 1. – P. 705–710.

155. Hang X. Applying both positive and negative selection to supervised learning for anomaly detection / X. Hang, H. Dai // Proceedings of the Genetic and Evolutionary Computation Conference (GECOO'05). – Washington, D.C., USA, 2005. – 25–29 June. – P. 345–352.

156. Tan K. The application of neural networks to Unix computer security / K. Tan // Proceedings of IEEE International Conference on Neural Networks. – Perth, WA, Australia, 1995. – Nov/Dec. – Vol. 1. – P. 476–481.

157. Файнзильберг Л. С. Гарантированная оценка эффективности диагностических тестов на основе усиленного ROC-анализа / Л.С. Файнзильберг, Т.Н. Жук // Управляющие системы и машины: информационные технологии : международный научный журнал. – 2009. – № 5. – С. 3–13.

158. Metz C.E. Fundamental ROC analysis / C.E. Metz // Progress in Medical Physics and Psychophysics Handbook of Medical Imaging. – Bellingham, WA, 2000. – Vol. 1. – P. 754–769.

159. Fawcett T. Using Rule Sets to Maximize ROC Performance / T. Fawcett // Proceedings of the IEEE International Conference on Data Mining (ICDM–2001). – Los Alamitos, CA, 2001. – P. 131–138.

160. Ferri C. Volume Under the ROC Surface for Multiclass Problems. Exact Computation and Evaluation of Approximations / C. Ferri, J. Hernández-Orallo, M.A. Salido // Techn. Rep. DSIC. – Univ. Politèc. València, 2003. – P. 36–43.

161. Green D.M., Swets J.A. Signal detection theory and psychophysics. – New York: John Wiley and Sons Inc., 1966. – 328 p.

162. Egan J.P. Signal detection theory and ROC analysis / J.P. Egan // New York: Acad. Press, 1975. – 386 p.

163. Swets J.A. Measuring the accuracy of diagnostic systems / J.A. Swets // Science. – 1988. – Vol. 240. – P. 1285–1292.

164. Spackman K.A. Signal detection theory: Valuable tools for evaluating inductive learning / K.A. Spackman // Proceedings of the Sixth International Workshop on Machine Learning. – San Mateo, CA, 1989. – P. 160–163.

165. Fawcett T. ROC Graphs: Notes and Practical Considerations for Researchers / T. Fawcett // Kluwer Acad. Publ., 2004. – 38 p.

166. Flach P. Repairing concavities in ROC curves / P. Flach, S. Wu // Proceedings UK Workshop on Comp. Intel. – 2003. – P. 38–44.

167. Zweig M.H. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine / M.H. Zweig, G. Campbell // Clinical Chemistry. – 1993. – Vol. 39, № 4. – P. 561–577.

168. Youden W.J. Index for rating diagnostic tests / W.J. Youden // Cancer. – 1950. – Vol. 3. – P. 32–35.

169. Optimal Cut-point and Its Corresponding Youden Index to Discriminate Individuals Using Pooled Blood Samples / E.F. Schlisterman, N.J. Perkins, A. Liu [et al.] // Epidemiology. – 2005. – Vol. 16. – P. 73–81.

170. Van den Hout W.B. The area under an ROC curve with limited information / W.B. Van den Hout // Medical Decision Making. – 2003. – Vol. 23. – P. 160–166.

171. Hanley J.A. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve / J.A. Hanley, B.J. McNeil // Radiology. – 1982. – Vol. 143. – P. 29–36.

172. Hand D.J. A simple generalization of the area under the ROC curve to multiple class classification problems / D.J. Hand, R.J. Till // Machine Learning. – 2001. – Vol. 45, № 2. – P. 171–186.

173. Srinivasan A. Note on the Location of Optimal Classifiers in N-dimensional ROC Space / A. Srinivasan // Techn. Rep. PRG-TR-2-99. – Oxford Univ. Comp. Labor., 1999. – 260 p.

174. Provost F. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distribution / F. Provost, T. Fawcett // Proceedings of The Third Intern. Conf. on Knowledge Discovery and Data Mining (KDD–97). – AAAI Press, 1997. – P. 43–48.

175. Provost F. Robust Classification for Imprecise Environments / F. Provost, T. Fawcett // Machine Learning. – 2001. – Vol. 42, № 3. – P. 203–231.

176. Bettinger R. Cost-Sensitive Classifier Selection Using the ROC Convex Hull Method / R. Bettinger // Comp. Scie. and Statistics. – 2003. – Vol. 35. – P. 36–42.

177. Drummond C. Explicitly representing expected cost: an alternative to ROC representation / C. Drummond, R. Holte // Proceedings of Knowledge Discovery and Datamining. – 2000. – P. 198–207.

178. Drummond C. What ROC curves can't do (and cost curves can) / C. Drummond, R. Holte // ROCAI. – 2004. – P. 19–26.

179. Похибки першого і другого роду : [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Похибки_першого_і_другого_роду.

180. Логистическая регрессия и ROC-анализ - математический аппарат : [Електронний ресурс]. – Режим доступу: http://www.basegroup.ru/library/analysis/regression/logistic.

181. Sabhnani M. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context / M. Sabhnani, G. Serpen // Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003). – Las Vegas, 2003. – Vol. 1. – P. 209–215.

182. Madhuri Agravat. Computer intrusion detection by two objective fuzzy genetic algorithm / Madhuri Agravat, Udai Pratap Rao // Computer Science & Information Technology. – 2011. – Vol. 2. – P. 281–292.

183. Al-Enezi J.R. Artificial immune systems – models, algorithms and applications / J.R. Al-Enezi, M.F. Abbod, S. Alsharhan / Artificial Immune Systems. – 2010. – Vol. 3, № 2. – P. 118–131.

184. Saravanan K. An Efficient Detection Mechanism for Intrusion Detection Systems Using Rule Learning Method / K. Saravanan // International Journal of Computer and Electrical Engineering. – 2009. – Vol. 1, № 4. – P. 503–506.

185. Shimonski R.J. Sniffer Pro network optimization and troubleshooting handbook / R.Shimonski, W. Eaton, U. Khan // Syngress, 2002. – 560 p.

186. Головко В.А. Нейронные сети: обучение, организация, применение / В.А. Головко // Нейрокомпьютеры и их применение : учеб. пособие. – М., 2001. – 256 с.

187. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.: Вильямс, 2006. – 1104 с.

188. Kohonen T. The self organizing map / T. Kohonen // Proceedings oft he Institute of Electrical and E1ectronics Engineers. – 1990. – Vo1. 78. – P. 1464 – 1480.

189. Анализатор трафика // Википедия [Електронний ресурс]. – 2005. – Режим доступу: http://ru.wikipedia.org/wiki/Анализатор трафика.

190. Jolliffe I. Principal component analysis / I.T. Jolliffe. – Springer, 2010. – 516 p.

191. Shilpa Lakhina. Feature Reduction using Principal Component Analysis for Effective Anomaly–Based Intrusion Detection on NSL-KDD / Shilpa Lakhina, Sini Joseph, Bhupendra Verma // International Journal of Engineering Science and Technology. – 2010. – Vol. 2, № 6. – P. 1790–1799.

192. Hofmeyr S. Immunity by design: An artificial immune system / S. Hofmeyr, S. Forrest // Gecco. – 1999. – Vol. 2. – P. 1289–1296.

Додаток А

Типи мережевих атак

Таблиця А.1 – Типи мережевих атак

№	Тип атаки	Опис	Клас атаки
1	back	ця атака здійснюється проти apache Web-сервера, який блокується великим потоком запитів, що містять велике число символів (/) в описі URL. Намагаючись обробити ці запити, сервер не здатний обслужити інші нормальні запити.	DOS
2	buffer_ overflow	принцип даної атаки побудований на використанні програмних помилок, що дозволяють викликати порушення меж пам’яті і аварійно завершити додатки або виконати довільний бінарний код від імені користувача, під яким працювала вразлива програма. Якщо програма працює під обліковим записом адміністратора системи, то дана атака дозволить отримати повний контроль над комп’ютером жертви.	U2R
3	ftp_write	це атака, коли атакер створює rhost-файл для того, щоб зробити анонімну ftp-директорію доступною на запис і у результаті отримати доступ до системи. ftp_write X write to a file on an FTP server.	R2L
4	guess_ passwd	злом файлу паролів	R2L
5	imap	дозволяє користувачам одержувати їх електронні листи з поштового сервера. В останній рік було випущено програмне забезпечення сервера IMAP, в якому містилися помилки, що дозволяють віддаленому корирстувачу отримувати повний контроль над машиною. Ця вразливість дуже небезпечна, оскільки велику кількість поштових серверів використовують вразливе програмне забезпечення IMAP.	R2L
6	ipsweep	функція IPSweep, яка проглядає тільки активні хости/об’єкти в заданому діапазоні. IPSweep визначить хости/об’єкти, навіть в тому випадку, якщо вони не відповідають на пінг (блокуються запити icmp). Користувачі можуть вручну додати IP-адреса, які вони хочуть просканувати або використовувати IPSweep, щоб їх визначити.	Probe
7	land	при атаках Land, атакер посилає жертві пакет TCP SYN, де IP-адреса відправника і одержувача ідентичні. Такий пакет повністю блокує роботу системи жертви.	DOS
8	loadmodule	loadmodule використовується серверною програмою для завантаження двох динамічно підвантажуваних драйверів ядра в поточну завантажену систему і створення спеціальних пристроїв в /dev директорії. Із-за наявності помилки в програмі неавторизовані користувачі можуть отримати права доступу до локальної машини.	U2R

Продовження таблиці А.1

9	multihop	деякі системи виявлення вторгнень проводять моніторинг трафіку безпосередньо за межами роутера і слідкують за вхідним і вихідним трафіком тільки мережі в цілому. Сценарій multihop розроблявся спеціально для того, щоб перевірити, чи зможе така система виявити атаку коли атакер спочатку зламає внутрішню машину, а потім, використовуючи цю машину для атаки на решту частини мережі. Це дуже ефективний спосіб атаки, оскільки системи виявлення вторгнень проводять моніторинг трафіку тільки зовні, але не всередині.	R2L
10	neptune	атакер посилає потік SYN пакетів на певний порт комп'ютера-жертви. На протязі деякого короткого проміжку часу після того, як ці пакети були послані, інші користувачі не мають можливості отримати доступу до служб, що забезпечуються цим портом.	DOS
11	nmap	для перевірки вашої системи на вразливість використовується програма NMAP. NMAP - дуже популярна програма, яка використовується хакерами для сканування Internet. Вона працює під Unix і має багато конфігураційних опцій і використовує декілька трюків, щоб уникнути детектування системами, що виявляють вторгнення. NMAP не дозволяє хакерові вторгнутися в систему, але допускає отримання ним корисної інформації про конфігурацію системи і доступні послуги. Програма часто використовується як прелюдія серйознішої атаки.	Probe
12	perl	атаки, що використовують помилки виконання Perl. В результаті, будь-хто може отримати права доступу адміністратора.	U2R
13	phf	використання погано написаних скриптів CGI, що дозволяють виконувати команди на http сервері з привілейованим рівнем. Будь-які CGI програми з використанням таких CGI функцій як escape_shell_cmd() можуть бути вразливими для атак.	R2L
14	pod	у атаках Ping of Death атакер формує пакет, який містить більше 65,536 байт, що більше межі, визначеної в IP-протоколі. Цей пакет викликає різного роду руйнування в машині, яка його отримує, іноді це викликає rebooting.	DOS
15	portsweep	сканування доступних портів для прослушки портів. Надалі зазвичай використовується для пошуку специфічних сервісів.	Probe
16	rootkit	даний сценарій може бути розглянутий як продовження стандартного сценарію злому. Руткіти – це набір програм, які призначені для допомоги атакеру в отриманні доступу до віддаленої машини. Типовий руткіт містить сніфер, su та інші програми з backdoors, які допомагають в доступі, а також нові версії ps, netstat, і ls, які приховують факт «прослушки» і запускають приховані файли. Одного разу встановлений руткіт дозволяє атакеру багато раз отримувати вкрадену інформацію.	U2R

Продовження таблиці А.1

17	satan	збір різної інформації про віддалений комп'ютер або віддалені мережі для вивчення наявності і настройок таких мережевих сервісів як finger, NFS, NIS, ftp and tftp, rexd, statd та ін. На підставі отриманої інформації виявляються можливі вразливості. Вони можуть полягати в некоректно встановлених або настроєних мережевих сервісах, добре відомих «дірок».	Probe
18	smurf	при атаці «smurf» жертва перевантажується великим числом пакетів-відгуків ICMP. Атакер посилає величезне число ICMP «echo-запитів» по широкомовних адресах багатьох субмереж. Ці пакети містять IP-адрес жертви як адресу відправника. Кожна машина субмережі пошле ICMP відгук машині-жертві. Атаки smurf достатньо небезпечні, оскільки вони є розподіленими. Для протидії цим атакам слід заборонити вхід зовнішніх пакетів, IP-адрес призначення яких є широкомовним. Слід також не пропускати через шлюз зовнішні пакети з адресою відправника з LAN. Більш ефективним є блокування передачі в Інтернет пакетів з локальної мережі з адресами відправника, що невідповідають локальним адресам.	DOS
19	spy	інформаційні «колекціонери», періодично «сканують» комп'ютер-жертву для збору інформації. Можуть шукати конфіденційну інформацію, або, наприклад, читати особисті листи користувача. Роблять певні кроки для мінімізації можливості виявлення присутності.	R2L
20	teardrop	поки пакет подорожує від відправника до машини одержувача, він може бути роздільний на невеликі фрагменти. Атака Teardrop створює потік IP-фрагментів з надмірно великими значеннями поля зсув (offset). Машина місця призначення, яка намагається відновити ці фальсифіковані фрагменти може блокуватися або навіть здійснити операцію перезавантаження.	DOS
21	warezclient	безпосередньо пов'язаний з warezmaster. Після того, як зловмисник розмістив на FTP сервері нелегальне ПЗ, користувачі можу викачувати його із зламаного сервера.	R2L
22	warezmaster	ці експлойти асоціюються з FTP протоколом. Стандартно, користувач «гість» не має прав запису на FTP сервер. Атака пов'язана з використанням «дірок» в ПЗ FTP сервера. Атакер отримує доступ до сервера і завантажує на нього нелегальне ПЗ.	R2L

Додаток Б

Атрибути мережевого трафіку

Таблиця Б.1 – Атрибути мережевого трафіку

№	Параметр	Опис	Тип
Вбудовані атрибути (intrinsic attributes) Ці атрибути витягуються із зони заголовка мережевих пакетів			
1	duration	час роботи з'єднання (сек)	Continuous, integer
2	protocol_type	тип протоколу (TCP, UDP, ICMP)	symbolic
3	service	служба на стороні приймача (http, smtp, telnet... and other)	symbolic
4	flag	прапор помилки з'єднання (нормальне/помилкове) можливі прапори: SF, S0, S1, S2, S3, OTH, REJ, RSTO, RSTOS0, SH, RSTRH, SHR	symbolic
5	src_bytes	кількість байт від джерела до приймача одного з'єднання	Continuous, integer
6	dst_bytes	кількість байт від приймача до джерела одного з'єднання	Continuous, integer
7	land	1 якщо з'єднання з (в) той же самий порт (або IP адрес), 0 інакше	Symbolic, binary
8	wrong_fragment	кількість пакетів з неправильною контрольною сумою	Continuous, integer
9	urgent	кількість термінових пакетів. Терміновий пакет – це пакет з активованим бітом терміновості	Continuous, integer
Атрибути контенту (content attributes) Ці атрибути витягуються експертами із зони контенту мережевих пакетів			
10	hot	кількість «hot» індикаторів, таких як доступ до системної директорії, запуск і завершення програм	Continuous, integer
11	num_failed_logins	кількість невдалих спроб реєстрації в системі за з'єднання	Continuous, integer
12	logged_in	1 якщо реєстрація в системі пройшла вдало, інакше - 0	Symbolic, binary
13	num_compromised	кількість виникнень помилки «не знайдена» за з'єднання	Continuous, integer
14	root_shell	1 якщо отримана root-оболонка, 0 інакше	Continuous, binary
15	su_attempted	1 якщо була спроба виконати команду «Su root», інакше - 0	Continuous, binary
16	num_root	кількість операцій класифікованих як root за з'єднання	Continuous, integer
17	num_file_creations	число операцій створення файлів за з'єднання	Continuous, integer

Продовження таблиці Б.1

18	num_shells	кількість входів (login) користувачів-normal	Continuous, integer
19	num_access_files	кількість операцій контролю над файлами за з'єднання	Continuous, integer
20	num_outbound_ cmds	кількість команд відправлення в ftp-сеccії	Continuous, integer
21	is_host_login	1 якщо користувач реєструється як root або adm, інакше - 0	Symbolic, binary
22	is_guest_login	1 якщо користувач реєструється як guest, anonymous або visitor, інакше - 0	Symbolic, binary
Атрибути трафіку (Traffic attributes) Ці атрибути обчислюються виходячи з попередніх з'єднань. 9 + 10 атрибутів поділені на дві групи: 1- часовий трафік (time traffic). 2 – машинний трафік (machine traffic). Різниця між групами полягає в режимі вибору попереднього пакету.			
Time traffic attributes Даний час з'єднання – 2 секунди			
23	count	кількість з'єднань до цього ж IP-адресу	Continuous, integer
24	srv_count	кількість з'єднань до цього ж номера порту	Continuous, integer
25	serror_rate	відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в count (23)	Continuous, real
26	srv_serror_rate	відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в srv_count (24)	Continuous, real
27	rerror_rate	відсоток з'єднань з активованим прапором flag (4) REJ впродовж з'єднань відображених в count (23)	Continuous, real
28	srv_rerror_rate	відсоток з'єднань з активованим прапором flag (4) REJ впродовж з'єднань відображених в srv_count (24)	Continuous, real
29	same_srv_rate	відсоток з'єднань до такої ж служби впродовж з'єднань відображених в count (23)	Continuous, real
30	diff_srv_rate	відсоток з'єднань до інших служб впродовж з'єднань відображених в count (23)	Continuous, real
31	srv_diff_host_rate	відсоток з'єднань до різних віддалених машин впродовж з'єднань, вказаних в srv_count (24)	Continuous, real
Machine traffic attributes Дана кількість з'єднань – 100 з'єднань			
32	dst_host_count	кількість з'єднань до такого ж IP-адресу	Continuous integer
33	dst_host_srv_count	кількість з'єднань до такого ж номера порту	Continuous, integer
34	dst_host_same_srv_ rate	відсоток з'єднань до такої ж служби впродовж з'єднань, відображених в dst_host_count (32)	Continuous, real

Продовження таблиці Б.1

35	dst_host_diff_srv_ rate	відсоток з'єднань до різних служб впродовж з'єднань, відображених в dst_host_count (32)	Continuous, real
36	dst_host_same_src_ port_rate	відсоток з'єднань до такого ж джерела порту впродовж з'єднань, відображених в dst_host_srv_count (33)	Continuous, real
37	dst_host_srv_diff_ host_rate	відсоток з'єднань до різних віддалених машин впродовж з'єднань, відображених в dst_host_srv_count (33)	Continuous, real
38	dst_host_serror_rate	відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в dst_host_count (32)	Continuous, real
39	dst_host_srv_serror_ rate	відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в dst_host_srv_count (33)	Continuous, real
40	dst_host_rerror_rate	відсоток з'єднань з активованим прапором flag (4) REJ впродовж з'єднань відображених в dst_host_count (32)	Continuous, real
41	dst_host_srv_rerror_ rate	відсоток з'єднань з активованим прапором flag (4) REJ впродовж з'єднань відображених в dst_host_srv_count (33)	Continuous, real

Додаток В

Адаптація параметрів з’єднань

Таблиця В.1 – Перетворення символьного типу в цілочисельний параметра protocol_type

Значення параметра до перетворення	Значення параметра після перетворення
tcp	1
udp	2
icmp	3

Таблиця В.2 – Перетворення символьного типу в цілочисельний параметра flag

Значення параметра до перетворення	Значення параметра після перетворення
SF	1
S0	2
S1	3
S2	4
S3	5
OTH	6
REJ	7
RSTO	8
RSTOS0	9
SH	10
RSTRH	11
SHR	12

Таблиця В.3 – Перетворення в цілочисельний тип (множення на 100)

№	Параметр	Тип	Приклад (до)	Перетворення	Приклад (після)
1.	duration (час роботи з’єднання (сек))	integer	0	немає	0
2.	protocol_type (тип протоколу (tcp, udp, icmp))	symbolic	tcp	a > 1	1
3.	service (служба на стороні приймача (http, smtp, telnet... and other))	symbolic	http	a > 1	1
4.	flag (прапор помилки з’єднання (нормальне/помилкове) можливі прапори: sf, s0, s1, s2, s3, oth, rej, rsto, rstos0, sh, rstrh, shr)	symbolic	sf	a > 1	1
5.	src_bytes (кількість байт від джерела до приймача одного з’єднання)	integer	181	немає	181
6.	dst_bytes (кількість байт від приймача до джерела одного з’єднання)	integer	5450	немає	5450

Продовження таблиці В.3

7.	land (1 якщо з'єднання з (в) той же самий порт (або ip адреса), 0 інакше)	binary	0	немає	0
8.	wrong_fragment (кількість пакетів з неправильною контрольною сумою)	integer	0	немає	0
9.	urgent (кількість термінових пакетів. терміновий пакет – це пакет з активованим бітом терміновості)	integer	0	немає	0
10.	hot (кількість «hot» індикаторів, таких як доступ до системної директорії, запуск і завершення програм)	integer	0	немає	0
11.	num_failed_logins (кількість невдалих спроб реєстрації в системі за з'єднання)	integer	0	немає	0
12.	logged_in (1 якщо реєстрація в системі пройшла вдало, інакше - 0)	binary	1	немає	1
13.	num_compromised (кількість виникнень помилки «не знайдена» за з'єднання)	integer	0	немає	0
14.	root_shell (1 якщо отримана root-оболонка, 0 інакше)	binary	0	немає	0
15.	su_attempted (1 якщо була спроба виконати команду «Su root», інакше - 0)	binary	0	немає	0
16.	num_root (кількість операцій класифікованих як root за з'єднання)	integer	0	немає	0
17.	num_file_creations (число операцій створення файлів за з'єднання)	integer	0	немає	0
18.	num_shells (кількість входів (login) користувачів-normal)	integer	0	немає	0
19.	num_access_files (кількість операцій контролю над файлами за з'єднання)	integer	0	немає	0
20.	num_outbound_cmds (кількість команд відправлення в ftp-сесії)	integer	0	немає	0
21.	is_host_login (1 якщо користувач реєструється як root або adm, інакше - 0)	binary	0	немає	0
22.	is_guest_login (1 якщо користувач реєструється як guest, anonymous або visitor, інакше - 0)	binary	0	немає	0
23.	count (кількість з'єднань до цієї ж ip-адреси)	integer	8	немає	8
24.	srv_count (кількість з'єднань до цього ж номера порту)	integer	8	немає	8
25.	serror_rate (відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в count (23))	real	0,00	0,1 > 1	0
26.	srv_serror_rate (відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в srv_count (24))	real	0,00	0,1 > 1	0

Продовження таблиці В.3

27.	rerror_rate (відсоток з'єднань з активованим прапором flag (4) rej впродовж з'єднань відображених в count (23))	real	0,00	0,1 > 1	0
28.	srv_rerror_rate (відсоток з'єднань з активованим прапором flag (4) rej впродовж з'єднань відображених в srv_count (24))	real	0,00	0,1 > 1	0
29.	same_srv_rate (відсоток з'єднань до такої ж служби впродовж з'єднань відображених в count (23))	real	1,00	0,1 > 1	100
30.	diff_srv_rate (відсоток з'єднань до інших служб впродовж з'єднань відображених в count (23))	real	0,00	0,1 > 1	0
31.	srv_diff_host_rate (відсоток з'єднань до різних віддалених машин впродовж з'єднань, вказаних в srv_count (24))	real	0,00	0,1 > 1	0
32.	dst_host_count (кількість з'єднань до такої ж ip-адреси)	integer	9	немає	9
33.	dst_host_srv_count (кількість з'єднань до такого ж номера порту)	integer	9	немає	9
34.	dst_host_same_srv_rate (відсоток з'єднань до такої ж служби впродовж з'єднань, відображених в dst_host_count (32))	real	1,00	0,1 > 1	100
35.	dst_host_diff_srv_rate (відсоток з'єднань до різних служб впродовж з'єднань, відображених в dst_host_count (32))	real	0,00	0,1 > 1	0
36.	dst_host_same_src_port_rate (відсоток з'єднань до такого ж джерела порту впродовж з'єднань, відображених в dst_host_srv_count (33))	real	0,11	0,1 > 1	11
37.	dst_host_srv_diff_host_rate (відсоток з'єднань до різних віддалених машин впродовж з'єднань, відображених в dst_host_srv_count (33))	real	0,00	0,1 > 1	0
38.	dst_host_serror_rate (відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в dst_host_count (32))	real	0,00	0,1 > 1	0
39.	dst_host_srv_serror_rate (відсоток з'єднань з активованим прапором flag (4) s0, s1, s2 і s3 впродовж з'єднань відображених в dst_host_srv_count (33))	real	0,00	0,1 > 1	0
40.	dst_host_rerror_rate (відсоток з'єднань з активованим прапором flag (4) rej впродовж з'єднань відображених в dst_host_count (32))	real	0,00	0,1 > 1	0
41.	dst_host_srv_rerror_rate (відсоток з'єднань з активованим прапором flag (4) rej впродовж з'єднань відображених в dst_host_srv_count (33))	real	0,00	0,1 > 1	0

Додаток Г

Приклади векторів для навчання НМ

```
% створення масивів даних для різних типів атак для навчання і перевірки функціонування
детекторів

% нормальне з'єднання (30 з'єднань)
Normal = [ 0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0
0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0 0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0 0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 1 0 0 0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 1 0 0 0 0 0;
0 2 13 1 105 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0 0 0 0;
2 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 255 253 99 1 1 0 0 0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0 0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0 0 0 0;
0 2 13 1 105 147 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 253 99 1 0 0 0 0 0 0;
0 1 11 1 12 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 14 12 86 14 86 0 0 0 0 0;
0 1 2 1 551 329 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 24 11 46 12 4 0 0 0 0 0;
0 1 2 1 1968 335 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 4 0 0 0 0 100 0 100 34 17 50 9 3 0 0 0 0 0;
0 1 2 1 1203 327 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 3 0 0 0 0 100 0 100 44 27 61 7 2 0 0 0 0 0;
1 1 2 1 1912 329 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 3 0 0 0 0 100 0 100 54 35 65 6 2 0 0 0 0 0;
2 1 2 1 1553 329 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 3 0 0 0 0 100 0 100 64 45 70 5 2 0 0 0 0 0;
2 1 2 1 1324 329 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 74 55 74 4 1 0 0 0 0 0;
0 1 11 1 192 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 84 20 24 4 24 0 0 0 0 0;
0 1 11 1 2890 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 8 8 0 0 0 0 100 0 0 94 30 32 3 32 0 0 0 0 0;
2 1 2 1 1684 363 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 104 66 63 3 1 0 0 0 0 0;
0 2 4 1 39 99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 246 96 1 0 0 0 0 0 0;
0 2 4 1 52 118 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 247 97 1 0 0 0 0 0 0;
0 2 4 1 52 118 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 247 97 1 0 0 0 0 0 0;
0 2 4 1 52 118 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 246 96 1 0 0 0 0 0 0;
0 2 4 1 31 91 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 100 0 100 255 246 96 1 0 0 0 0 0 0;
0 2 4 1 31 91 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 246 96 1 0 0 0 0 0 0;
0 2 4 1 44 110 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 246 96 1 0 0 0 0 0 0;
0 2 4 1 44 110 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 5 0 0 0 0 100 0 60 255 246 96 1 0 0 0 0 0 0;
0 2 4 1 34 94 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 246 96 1 0 0 0 0 0 0];

% DOS (30 з'єднань)
Back = [ 0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 1 2 0 0 0 50 100 0 100 1 1 100 0 100 0
0 0 0 0;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 2 3 0 0 0 33 100 0 67 2 2 100 0 50 0 0 0 0 0;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 3 4 0 0 0 25 100 0 50 3 3 100 0 33 0 0 0 0 0;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 4 4 100 0 25 0 0 0 0 0;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 5 5 100 0 20 0 0 0 0 0;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 6 6 100 0 17 0 0 0 0 0;
0 1 1 11 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 25 25 100 0 0 7 7 100 0 14 0 0 0 14
14;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 20 20 100 0 0 8 8 100 0 12 0 0 0 12
12;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 20 20 100 0 0 9 9 100 0 11 0 0 0 11
11;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 6 0 0 17 17 100 0 0 10 10 100 0 10 0 0 0 10
10;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 7 7 0 0 14 14 100 0 0 11 11 100 0 9 0 0 0 9 9;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 6 0 0 17 17 100 0 0 12 12 100 0 8 0 0 0 8 8;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 13 13 100 0 8 0 0 0 8 8;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 6 0 0 0 0 100 0 0 14 14 100 0 7 0 0 0 7 7;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 15 15 100 0 7 0 0 0 7 7;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 16 16 100 0 6 0 0 0 6 6;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 17 17 100 0 6 0 0 0 6 6;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 18 18 100 0 6 0 0 0 6 6;
0 1 1 11 54060 7300 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 4 4 0 0 25 25 100 0 0 19 19 100 0 5 0 0 0 11
11;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 20 20 100 0 0 20 20 100 0 5 0 0 0 10
10;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 25 25 100 0 0 21 21 100 0 5 0 0 0 10
10;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 25 25 100 0 0 22 22 100 0 5 0 0 0 9 9;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 20 20 100 0 0 23 23 100 0 4 0 0 0 9 9;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 24 24 100 0 4 0 0 0 8 8;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 25 25 100 0 4 0 0 0 8 8;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 26 26 100 0 4 0 0 0 8 8;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 27 27 100 0 4 0 0 0 7 7;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 28 28 100 0 4 0 0 0 7 7;
```

```
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 29 29 100 0 3 0 0 0 7 7;
0 1 1 1 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 30 30 100 0 3 0 0 0 7 7];

%U2R (30 з'єднань)
Buffer_overflow = [184 1 5 1 1511 2957 0 0 0 3 0 1 2 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 3
100 0 100 67 0 0 0 0;
305 1 5 1 1735 2766 0 0 0 3 0 1 2 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 4 100 0 50 50 0 0 0 0;
150 1 5 1 1587 6707 0 0 0 1 0 1 3 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
60 1 5 1 2328 4551 0 0 0 3 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
158 1 5 1 1567 3095 0 0 0 3 0 1 4 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 2 100 0 50 0 0 0 0 0;
113 1 5 1 6274 16771 0 0 0 5 0 1 2 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
53 1 5 1 2628 3860 0 0 0 3 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 2 100 0 50 0 0 0 0 0;
0 1 11 1 0 5690 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
0 1 11 1 0 5828 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 2 100 0 100 0 0 0 0 0;
0 1 11 1 0 5020 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 3 3 100 0 100 0 0 0 0 0;
0 1 11 1 0 2072 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 4 4 100 0 100 0 0 0 0 0;
7 1 11 1 226 698 0 0 0 4 0 1 0 0 0 0 4 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
169 1 5 1 1567 2857 0 0 0 3 0 1 4 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
179 1 5 1 1559 2855 0 0 0 3 0 1 4 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 2 100 0 50 0 0 0 0 0;
49 1 5 1 2402 3939 0 0 0 4 0 1 2 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 2 100 0 100 100 0 0 0 0;
290 1 5 1 415 70529 0 0 0 3 0 1 4 0 0 4 4 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
31 1 5 1 137 1351 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 2 100 0 50 0 0 0 0 0;
0 1 11 1 0 5696 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 81 100 0 100 2 0 0 0 0;
0 1 11 1 0 5928 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 82 100 0 100 2 0 0 0 0;
0 1 11 1 0 5020 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 3 83 100 0 100 2 0 0 0 0;
0 1 11 1 0 2072 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 4 84 100 0 100 2 0 0 0 0;
31 1 5 1 2402 3814 0 0 0 3 0 1 2 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 2 100 0 50 0 0 0 0 0;
33 1 5 1 2402 3815 0 0 0 3 0 1 2 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 3 100 0 33 0 0 0 0 0;
162 1 5 1 1567 2738 0 0 0 3 0 1 4 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 4 4 100 0 25 0 0 0 0 0;
127 1 5 1 1567 2736 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 83 1 99 0 0 0 1 8 0 5 5 100 0 20 0 0 0 0 0;
321 1 5 8 1506 1887 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 151 1 99 0 1 100 1 6 0 6 6 100 0 17 0 0 0 17
17;
45 1 5 1 2336 4201 0 0 0 3 0 1 1 1 0 0 0 0 0 0 0 0 2 1 0 0 50 0 50 100 0 7 7 100 0 14 0 0 0 14
14;
176 1 5 1 1559 2732 0 0 0 3 0 1 4 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 8 8 100 0 12 0 0 0 12 12;
61 1 5 1 2336 4194 0 0 0 3 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 9 9 100 0 11 0 0 0 11 11;
47 1 5 1 2402 3816 0 0 0 3 0 1 2 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 10 10 100 0 10 0 0 0 10
10];

%R2L (8 з'єднань)
Ftp_write = [26 1 11 1 116 451 0 0 0 2 0 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 100 0 0 1 1 100 0 100
0 0 0 0 0;
134 1 12 1 100 39445 0 0 2 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 100 0 0 2 1 50 100 50 0 0 0 0 0;
0 1 11 1 613 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 2 0 0 0 0 100 0 100 1 84 100 0 100 2 0 0 0 0;
0 1 11 1 0 5 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 85 100 0 100 2 0 0 0 0;
32 1 11 1 104 449 0 0 0 2 0 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
67 1 12 1 157 2703 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 1 50 100 50 0 0 0 0 0;
0 1 11 1 676 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 4 100 0 100 50 0 0 0 0;
0 1 11 1 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 5 100 0 100 40 0 0 0 0];

%R2L (30 з'єднань)
Guess_passwd = [23 1 5 1 104 276 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 2 100 0
100 100 0 0 0 0;
60 1 5 5 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 1 1 100 0 100 0 100 100
0 0;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 50 50 50 50 100 0 0 2 2 100 0 50 0 50 50 50
50;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 3 3 100 0 33 0 33 33 67
67;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 4 4 100 0 25 0 25 25 75
75;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 5 5 100 0 20 0 20 20 80
80;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 6 6 100 0 17 0 17 17 83
83;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 7 7 100 0 14 0 14 14 86
86;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 8 8 100 0 12 0 12 12 88
88;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 9 9 100 0 11 0 11 11 89
89;
0 1 5 8 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 10 10 100 0 10 0 10 10 90
90;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 11 11 100 0 9 0 9 9 91
91;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 12 12 100 0 8 0 8 8 92
92;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 13 13 100 0 8 0 8 8 92
92;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 14 14 100 0 7 0 7 7 93
```

```
93;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 15 15 100 0 7 0 7 7 93
93;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 16 16 100 0 6 0 6 6 94
94;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 17 17 100 0 6 0 6 6 94
94;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 18 18 100 0 6 0 6 6 94
94;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 19 19 100 0 5 0 5 5 95
95;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 20 20 100 0 5 0 5 5 95
95;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 21 21 100 0 5 0 5 5 95
95;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 22 22 100 0 5 0 5 5 95
95;
60 1 5 5 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 50 50 50 50 100 0 0 23 23 100 0 4 0 9 9 91
91;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 24 24 100 0 4 0 8 8 92
92;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 25 25 100 0 4 0 8 8 92
92;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 26 26 100 0 4 0 8 8 92
92;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 27 27 100 0 4 0 7 7 93
93;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 28 28 100 0 4 0 7 7 93
93;
0 1 5 8 126 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 100 100 100 0 0 29 29 100 0 3 0 7 7 93
93] ;

%R2L (12 з'єднань)
Imap = [ 31 1 14 1 1345 10036 0 0 0 0 0 1 16 0 0 16 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 1 0 0
0 0 3 0;
0 1 14 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 160 100 58 0 3 100 0 28 1 1 100 0 100 0 100 100 0
0;
0 1 14 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 161 100 58 0 3 100 0 28 2 2 100 0 100 0 100 100 0
0;
0 1 14 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 162 67 58 0 3 100 0 28 3 3 100 0 100 0 67 67 0 0;
0 1 14 1 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 163 50 58 0 3 100 0 28 4 4 100 0 100 0 50 50 0 0;
41 1 14 1 1334 162 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 5 5 100 0 20 0 40 40 0 0;
0 1 14 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 6 6 100 0 17 0 50 50 0 0;
0 1 14 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 122 100 76 0 4 100 0 37 7 7 100 0 71 0 57 57 0 0;
0 1 14 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 123 100 76 0 4 100 0 37 8 8 100 0 75 0 62 62 0 0;
0 1 14 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 124 67 76 0 4 100 0 36 9 9 100 0 78 0 56 56 0 0;
0 1 14 1 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 125 50 75 0 4 100 0 36 10 10 100 0 80 0 50 50 0 0;
0 1 14 3 1492 649186 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 11 11 100 0 9 0 55
55 0 0] ;

%Probe (30 з'єднань)
Ipsweep = [ 0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 9 1 11 100 11 0 0
0 89 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 19 1 5 100 5 0 0 0 95 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 29 1 3 100 3 0 0 0 97 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 39 1 3 100 3 0 0 0 95 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 49 1 2 100 2 0 0 0 96 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 59 1 2 100 2 0 0 0 95 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 69 1 1 100 1 0 0 0 94 100;
0 1 2 1 0 91 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 6 51 17 100 17 4 0 0 83 0;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 16 2 6 100 6 100 0 0 94 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 26 2 4 100 4 100 0 0 92 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 36 2 3 100 3 100 0 0 92 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 46 2 2 100 2 100 0 0 93 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 56 2 2 100 2 100 0 0 93 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 66 2 2 100 2 100 0 0 92 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 8 1 12 100 12 0 0 0 75 100;
0 1 15 1 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 18 1 6 100 6 0 0 0 83 0;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 28 1 4 100 4 0 0 0 89 100;
0 1 2 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 38 1 3 100 3 0 0 0 89 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 48 1 2 100 2 0 0 0 92 100;
0 1 16 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 58 1 2 100 2 0 0 0 91 100;
0 1 17 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 68 1 1 100 1 0 0 0 91 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 7 2 14 100 14 100 0 0 100
100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 17 2 6 100 6 100 0 0 100
100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 27 2 4 100 4 100 0 0 96 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 37 2 3 100 3 100 0 0 97 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 47 2 2 100 2 100 0 0 98 100;
```

```
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 57 2 2 100 2 100 0 0 98 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 67 2 1 100 1 100 0 0 97 100;
2 1 11 8 0 133 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 2 2 50 100 50 100 0 0 100
100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 100 100 100 0 0 12 2 8 100 8 100 0 0 92
100];

%DoS (21 з'єднань)
Land = [ 0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 1 8 100 0 100 38 100
12 0 0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 1 100 0 100 0 100 100 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 1 6 100 0 100 33 100 17 0 0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 12 9 8 17 8 33 58 11 0 0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 1 100 0 100 0 100 100 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 2 100 0 100 100 100 100 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 3 100 0 100 100 100 100 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 4 100 0 100 100 100 100 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 5 100 0 100 100 100 100 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 6 100 0 100 33 100 17 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 1 8 100 0 100 25 100 12 0 0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 6 100 0 100 33 100 17 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 2 100 0 100 100 100 50 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 3 100 0 100 100 100 67 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 4 100 0 100 100 100 75 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 5 100 0 100 100 100 80 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 1 6 100 0 100 100 100 83 0
0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 78 2 100 100 0 0 1 8 100 1 6 100 0 100 50 100 17 0 0;
0 1 5 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 1 1 100 0 100 0 100 100 0 0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 100 100 0 0 100 0 100 15 1 7 20 7 0 7 100 7 0;
0 1 3 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 2 33 100 67 0 33 100 100 16 2 12 19 12 0 12 100 6
0];

%U2R (9 з'єднань)
Loadmodule = [ 79 1 5 1 281 1301 0 0 0 2 0 1 1 1 0 0 4 2 0 0 0 0 1 1 0 0 0 0 100 0 0 1 10 100 0
100 30 0 0 0 10;
103 1 5 1 302 8876 0 0 0 2 0 1 4 1 0 3 4 2 1 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
0 1 11 1 0 5921 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 2 1 0 0 0 0 50 100 0 1 3 100 0 100 67 0 0 0 0;
0 1 11 1 0 5014 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 3 2 0 0 0 0 67 67 0 2 4 100 0 100 50 0 0 0 0;
0 1 11 1 0 2072 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 4 3 0 0 0 0 75 50 0 3 5 100 0 100 40 0 0 0 0;
7 1 11 1 230 644 0 0 0 4 0 1 0 0 0 0 4 0 0 0 0 0 1 1 0 0 0 0 100 0 0 4 1 25 75 25 0 0 0 0 0;
31 1 5 1 142 1278 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 5 3 60 60 20 0 0 0 0 0;
21 1 5 1 135 1290 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 6 4 67 50 17 0 0 0 0 0;
85 1 5 1 277 693 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0];

%R2L (7 з'єднань)
Multihop = [ 192 1 11 1 119 426 0 0 0 2 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 100 0 0 255 1 0 1 0 0
0 0 4 0;
179 1 11 1 87 319 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 100 0 0 255 2 1 1 0 0 0 0 4 0;
0 1 11 1 866 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
0 1 11 1 0 467968 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 2 2 100 0 100 0 0 0 0 0;
1 1 11 1 0 988002 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 100 0 100 3 3 100 0 100 0 0 0 0 0;
198 1 5 1 562 9139 0 0 0 3 0 1 22 1 0 39 4 2 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
718 1 5 1 1412 25260 0 0 0 15 0 1 38 1 0 54 4 1 2 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0
0];

%DoS (30 з'єднань)
Neptune = [ 0 1 5 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 5 83 100 0 0 83 33 0 5 6 100 0 20 33 100
83 0 0;
0 1 5 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 15 94 100 0 0 94 12 0 15 16 100 0 7 12 100 94 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 63 1 100 100 0 0 2 8 0 1 1 100 0 100 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 110 8 100 100 0 0 7 6 0 11 8 73 27 9 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 119 17 100 100 0 0 14 6 0 21 17 81 14 5 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 128 6 100 100 0 0 5 6 0 31 6 19 13 3 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 138 16 100 100 0 0 12 6 0 41 16 39 10 2 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 105 6 100 100 0 0 6 6 0 51 5 10 10 2 0 100 100 0 0;
```

```
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 114 14 100 100 0 0 12 6 0 61 13 21 10 2 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 123 3 100 100 0 0 2 7 0 71 3 4 10 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 132 12 100 100 0 0 9 6 0 81 12 15 9 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 141 2 100 100 0 0 1 6 0 91 2 2 8 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 150 11 100 100 0 0 7 5 0 101 11 11 7 1 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 109 19 100 100 0 0 17 6 0 111 19 17 7 1 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 119 10 100 100 0 0 8 6 0 121 10 8 7 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 128 18 100 100 0 0 14 6 0 131 18 14 7 1 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 74 12 100 100 0 0 16 7 0 141 12 9 6 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 97 14 100 100 0 0 14 6 0 151 14 9 7 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 106 7 100 100 0 0 7 6 0 161 7 4 6 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 115 16 100 100 0 0 14 5 0 171 16 9 6 1 0 100 100 0
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 74 4 100 100 0 0 5 8 0 181 4 2 7 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 83 13 100 100 0 0 16 7 0 191 13 7 7 1 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 92 3 100 100 0 0 3 7 0 201 3 1 6 0 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 62 3 100 100 0 0 5 8 0 211 3 1 7 0 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 71 12 100 100 0 0 17 7 0 221 12 5 6 0 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 80 11 100 100 0 0 14 6 0 231 11 5 6 0 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 61 19 100 100 0 0 31 7 0 241 19 8 6 0 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 70 8 100 100 0 0 11 7 0 251 8 3 6 0 0 100 100 0 0;
0 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 52 4 100 75 0 0 6 10 50 255 3 1 6 0 0 100 100 0 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 58 9 100 100 0 0 16 7 0 255 9 4 6 0 0 100 100 0 0];

%Probe (30 з'єднань)
Nmap = [ 0 3 6 1 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 4 2 50 75 50 0 0 0 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 100 100 0 0 50 100 0 14 1 7 71 50 0 50 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 24 1 4 75 67 0 67 100 0 0;
0 1 5 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 34 1 3 74 76 0 76 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 44 1 2 77 82 0 82 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 54 1 2 81 85 0 85 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 64 1 2 84 88 0 88 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 74 1 1 86 89 0 89 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 84 1 1 88 90 0 90 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 94 1 1 89 91 0 91 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 104 1 1 90 92 0 92 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 114 1 1 91 93 0 93 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 124 1 1 92 94 0 94 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 134 1 1 93 94 0 94 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 144 1 1 93 94 0 94 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 154 1 1 94 95 0 95 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 164 1 1 94 95 0 95 100 0 0;
0 1 18 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 174 1 1 94 95 0 95 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 184 1 1 95 96 0 96 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 194 1 1 95 96 0 96 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 204 1 0 95 96 0 96 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 214 1 0 95 96 0 96 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 224 1 0 96 96 0 96 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 234 1 0 96 97 0 97 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 244 1 0 95 97 0 97 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 254 1 0 96 97 0 97 100 0 0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 255 1 0 98 100 0 100 100 0
0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 255 1 0 99 100 0 100 100 0
0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 255 1 0 100 100 0 100 100 0
0;
0 1 13 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 100 100 0 0 100 0 0 255 1 0 100 100 0 100 100 0
0] ;

%U2R (3 з'єднань)
Perl = [ 25 1 5 1 269 2333 0 0 0 0 0 1 0 1 0 2 2 1 0 0 0 0 1 1 0 0 0 0 100 0 0 69 2 3 6 1 0 0 0 0
0;
54 1 5 1 260 2635 0 0 0 0 0 1 0 1 0 2 2 1 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 1 0 0 0 0 0 0;
45 1 5 1 268 2364 0 0 0 0 0 1 0 1 0 2 2 1 0 0 0 0 1 1 0 0 0 0 100 0 0 255 2 1 2 0 0 0 0 69 0] ;

%R2L (4 з'єднань)
Phf = [ 0 1 1 1 51 8127 0 0 0 2 0 1 0 1 0 0 0 0 1 0 0 0 1 2 0 0 0 50 100 0 100 255 255 100 0 0 0 0
0 0 0;
12 1 1 1 51 8127 0 0 0 2 0 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 100 0 0 255 246 96 1 0 0 0 0 0 0;
0 1 1 1 51 8127 0 0 0 2 0 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 100 0 0 255 241 95 1 0 0 0 0 0 0;
6 1 1 1 51 8127 0 0 0 2 0 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 100 0 0 255 249 98 1 0 0 0 0 0 0] ;

%DoS (30 з'єднань)
Pod = [ 0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 85 1 1 5 1 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 86 2 2 5 2 0 0 0 0 0;
```

```
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 87 3 3 5 3 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 88 4 5 5 5 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 89 5 6 4 6 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 6 0 0 0 0 100 0 0 90 6 7 4 7 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 7 0 0 0 0 100 0 0 91 7 8 4 8 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 8 0 0 0 0 100 0 0 92 8 9 4 9 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 9 0 0 0 0 100 0 0 93 9 10 4 10 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 10 0 0 0 0 100 0 0 94 10 11 4 11 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 11 0 0 0 0 100 0 0 95 11 12 4 12 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 12 0 0 0 0 100 0 0 96 12 12 4 12 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 13 0 0 0 0 100 0 0 97 13 13 4 13 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 14 0 0 0 0 100 0 0 98 14 14 4 14 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 15 0 0 0 0 100 0 0 99 15 15 4 15 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 16 0 0 0 0 100 0 0 100 16 16 4 16 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 17 0 0 0 0 100 0 0 101 17 17 4 17 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 18 0 0 0 0 100 0 0 102 18 18 4 18 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 19 0 0 0 0 100 0 0 103 19 18 4 18 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 20 0 0 0 0 100 0 0 104 20 19 4 19 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 2 0 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 255 2 1 2 1 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 255 3 1 2 1 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 255 4 2 2 2 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 255 5 2 2 2 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 6 0 0 0 0 100 0 0 255 6 2 2 2 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 7 0 0 0 0 100 0 0 255 7 3 2 3 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 8 0 0 0 0 100 0 0 255 8 3 2 3 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 9 0 0 0 0 100 0 0 255 9 4 2 4 0 0 0 0 0;
0 3 9 1 1480 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 10 0 0 0 0 100 0 0 255 10 4 2 4 0 0 0 0 0];

%Probe (29 з'єднань)
Portsweep = [ 1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 2 0 0 100 100 25 50 0 178 2 1 4 4 0
1 0 32 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 2 0 0 100 100 11 50 0 188 2 1 6 10 0 1 0 36 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28 1 0 0 100 100 4 54 0 198 1 1 9 14 0 1 0 39 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 1 0 0 100 100 3 55 0 208 1 0 11 18 0 1 0 42 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 2 0 0 100 100 4 54 0 218 2 1 12 22 0 1 0 44 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 58 2 0 0 100 100 3 53 0 228 2 1 14 25 0 1 0 47 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 68 1 0 0 100 100 1 54 0 238 1 0 16 29 0 1 0 49 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 78 2 0 0 100 100 3 54 0 248 2 1 17 31 0 1 0 51 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 88 2 0 0 100 100 2 53 0 255 2 1 19 35 0 0 0 53 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 98 2 0 0 100 100 2 55 0 255 2 1 22 38 0 0 0 53 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 108 1 0 0 100 100 1 56 0 255 1 0 24 42 0 0 0 53 100;
1 1 13 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 118 2 0 0 100 100 2 56 0 255 2 1 26 46 0 0 0 53
100;
1 1 16 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 128 2 0 0 100 100 2 56 0 255 2 1 28 50 0 0 0 53
100;
1 1 13 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 138 2 4 50 96 50 1 57 0 255 2 1 31 54 0 2 50 52 50;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 148 2 8 100 92 0 1 58 0 255 2 1 34 58 0 5 100 53 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 158 2 12 100 88 0 1 58 0 255 2 1 36 62 0 7 100 55 0;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 168 1 12 0 88 100 1 58 0 255 1 0 38 66 0 8 0 58 100;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 1 71 100 29 0 14 57 0 255 1 0 2 3 0 2 100 1 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 1 76 100 24 0 6 47 0 255 1 0 4 7 0 5 100 2 0;
1 1 13 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 3 78 67 22 33 11 41 0 255 3 1 5 11 0 8 67 2 33;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 2 81 100 19 0 5 41 0 255 2 1 6 15 0 12 100 3 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 47 2 81 100 19 0 4 40 0 255 2 1 8 18 0 15 100 4 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 67 1 79 100 21 0 1 39 0 255 1 0 10 26 0 21 100 5 0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 77 2 79 100 21 0 3 38 0 255 2 1 11 30 0 24 100 6 0;
1 1 13 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 87 3 77 67 23 33 3 37 0 255 3 1 13 34 0 26 67 8 33;
1 1 13 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 97 2 75 50 25 50 2 37 0 255 2 1 14 38 0 29 50 9 50;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 107 1 75 100 25 0 1 37 0 255 1 0 16 42 0 31 100 11
0;
0 1 13 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 117 2 74 100 26 0 2 37 0 255 2 1 17 46 0 34 100 12
0;
1 1 13 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 127 3 73 67 27 33 2 36 0 255 3 1 18 50 0 36 67 13
33];

%U2R (10 з'єднань)
Rootkit = [ 60 1 5 1 86 183 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 2 0 0 0 0
0 0;
60 1 5 1 90 233 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 2 1 2 0 0 0 0 0 0;
708 1 5 1 1727 24080 0 0 0 0 0 1 6 0 0 7 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 3 1 2 0 0 0 0 0 0;
21 1 11 1 89 345 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 2 0 0 0 0 0 0;
98 1 5 1 621 8356 0 0 1 1 0 1 5 1 0 14 1 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 4 2 2 0 0 0 0 0 0;
0 1 11 1 0 5636 0 0 0 0 0 0 1 2 0 0 2 0 0 0 0 0 0 1 1 0 0 0 0 0 100 0 0 1 41 100 0 100 10 0 0 0 0;
0 1 5 1 294 3929 0 0 0 0 0 0 1 0 1 0 4 1 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 4 2 2 0 0 0 25 73 25;
0 2 10 1 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 2 0 0 0 0 0 0;
0 2 10 1 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
0 2 10 1 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 2 100 0 50 0 0 0 0 0];

%Probe (30 з'єднань)
```

```
Satan = [ 0 2 4 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 1 0 0 0 0 17 67 0 150 1 1 6 3 0 1 0 0 0;
3 1 5 1 54 114 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 1 0 0 29 0 14 71 0 160 29 18 8 1 0 1 3 1 0;
0 3 6 1 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 1 0 2 0 0 0 0 0 0;
1 1 3 1 6 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 3 0 0 0 0 75 50 0 255 3 1 2 0 0 0 0 0 0;
0 1 19 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 0 0 33 100 33 100 0 255 1 0 4 0 0 0 0 0 100;
0 1 20 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 1 0 0 89 100 11 100 0 255 1 0 7 0 0 0 0 4 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 1 0 0 89 100 5 100 0 255 1 0 11 0 0 0 0 7 100;
0 1 1 4 6 144 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 29 1 3 100 90 0 3 100 0 255 206 81 15 0 0 0 0 11 0;
2 1 21 1 6 77 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 39 1 3 0 87 0 3 100 0 255 2 1 18 0 0 0 0 14 0;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 1 2 0 90 100 2 100 0 255 1 0 22 0 0 0 0 18 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 59 1 2 0 92 100 2 100 0 255 1 0 26 0 0 0 0 22 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 69 1 1 0 93 100 1 100 0 255 1 0 30 0 0 0 0 26 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 79 1 1 0 92 100 1 100 0 255 1 0 34 0 0 0 0 29 100;
0 1 11 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 89 1 1 0 90 100 1 100 0 255 1 0 36 0 0 0 0 32 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 99 1 1 0 91 100 1 100 0 255 1 0 40 0 0 0 0 36 100;
0 1 22 1 44 192 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 109 1 1 0 91 0 1 100 0 255 1 0 44 0 0 0 0 40 0;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 120 1 1 0 92 100 1 100 0 255 1 0 48 0 0 0 0 44 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 130 1 1 0 92 100 1 100 0 255 1 0 52 0 0 0 0 47 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 140 1 4 0 88 100 1 99 0 255 1 0 56 0 0 2 0 49 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 150 1 4 0 88 100 1 99 0 255 1 0 60 0 0 2 0 52 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 160 1 4 0 89 100 1 99 0 255 1 0 64 0 0 2 0 56 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 170 1 4 0 89 100 1 99 0 255 1 0 67 0 0 2 0 60 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 180 1 3 0 90 100 1 99 0 255 1 0 71 0 0 2 0 64 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 190 1 7 0 87 100 1 99 0 255 1 0 75 0 0 5 0 65 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 200 1 6 0 88 100 0 100 0 255 1 0 79 0 0 5 0 69 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 210 1 6 0 88 100 0 100 0 255 1 0 83 0 0 5 0 73 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 220 1 6 0 88 100 0 100 0 255 1 0 87 0 0 5 0 76 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 230 1 6 0 89 100 0 100 0 255 1 0 91 0 0 5 0 80 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 240 1 9 0 86 100 0 100 0 255 1 0 95 0 0 8 0 81 100;
0 1 13 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 250 1 8 0 86 100 0 100 0 255 1 0 98 0 0 8 0 85 100];

%DoS (30 з’єднань)
Smurf = [ 0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 316 316 0 0 0 0 100 0 0 148 3 2 2 2 0 0 0
0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 158 13 8 2 8 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 168 23 14 2 14 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 510 510 0 0 0 0 100 0 0 178 33 19 2 19 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 509 509 0 0 0 0 100 0 0 188 43 23 2 23 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 198 53 27 2 27 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 208 63 30 1 30 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 509 509 0 0 0 0 100 0 0 218 73 33 1 33 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 228 83 36 1 36 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 238 93 39 1 39 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 248 103 42 1 42 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 113 44 1 44 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 123 48 1 48 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 133 52 1 52 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 143 56 1 56 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 153 60 1 60 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 163 64 1 64 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 173 68 1 68 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 183 72 1 72 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 193 76 1 76 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 203 80 1 80 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 213 84 1 84 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 223 87 1 87 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 233 91 1 91 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 243 95 1 95 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 253 99 1 99 0 0 0 0 0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 255 100 0 100 0 0 0 0
0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 255 100 0 100 0 0 0 0
0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 255 100 0 100 0 0 0 0
0;
0 3 9 1 1032 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 100 0 0 255 255 100 0 100 0 0 0 0
0];

%R2L (2 з’єднань)
Spy = [ 337 1 5 1 237 1540 0 0 0 0 0 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 100 0 0 255 47 18 2 0 0 22
32 0 0;
299 1 5 1 112 847 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 255 48 19 2 0 0 22 31 0 0];

%DoS (30 з’єднань)
Teardrop = [ 0 2 13 1 28 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 73 1 1 5 1 0 0 0 0
0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 74 2 3 5 3 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 75 3 4 5 4 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0 0 100 0 0 76 4 5 5 5 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0 0 100 0 0 77 5 6 5 6 0 0 0 0 0;
```

```
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 6 0 0 0 0 100 0 0 78 6 8 5 8 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 7 0 0 0 0 100 0 0 79 7 9 5 9 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 8 0 0 0 0 100 0 0 80 8 10 5 10 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 9 0 0 0 0 100 0 0 81 9 11 5 11 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 10 0 0 0 0 100 0 0 82 10 12 5 12 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 11 0 0 0 0 100 0 0 83 11 13 5 13 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 12 0 0 0 0 100 0 0 84 12 14 5 14 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 13 0 0 0 0 100 0 0 85 13 15 5 15 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 14 0 0 0 0 100 0 0 86 14 16 5 16 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 15 0 0 0 0 100 0 0 87 15 17 5 17 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 16 0 0 0 0 100 0 0 88 16 18 5 18 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 17 0 0 0 0 100 0 0 89 17 19 4 19 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 18 0 0 0 0 100 0 0 90 18 20 4 20 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 19 0 0 0 0 100 0 0 91 19 21 4 21 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 20 0 0 0 0 100 0 0 92 20 22 4 22 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 21 0 0 0 0 100 0 0 93 21 23 4 23 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 22 0 0 0 0 100 0 0 94 22 23 4 23 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 23 0 0 0 0 100 0 0 95 23 24 4 24 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 24 0 0 0 0 100 0 0 96 24 25 4 25 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 25 0 0 0 0 100 0 0 97 25 26 4 26 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 26 0 0 0 0 100 0 0 98 26 27 4 27 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 27 0 0 0 0 100 0 0 99 27 27 4 27 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28 28 0 0 0 0 100 0 0 100 28 28 4 28 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 29 0 0 0 0 100 0 0 101 29 29 4 29 0 0 0 0 0;
0 2 13 1 28 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 30 0 0 0 0 100 0 0 102 30 29 4 29 0 0 0 0 0];

%R2L (30 з'єднань)
Warezclient = [0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0
0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 2 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 3 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 4 4 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 2 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 3 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 4 4 100 0 100 0 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 5 100 0 100 40 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 6 100 0 100 33 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 7 100 0 100 29 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 4 8 100 0 100 25 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 9 100 0 100 33 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 10 100 0 100 30 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 11 100 0 100 27 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 4 12 100 0 100 25 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 13 100 0 100 31 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 14 100 0 100 29 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 15 100 0 100 27 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 4 15 100 0 100 27 0 0 0 0;
0 1 11 1 246 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 13 100 0 100 31 0 0 0 0;
12 1 11 1 7045 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 14 100 0 100 29 0 0 0 0;
4 1 11 1 832 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 3 15 100 0 100 27 0 0 0 0;
5057 1 11 1 5133876 0 0 0 0 4 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 4 16 100 0 100 25 0 0 0
0;
0 1 11 1 30 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 5 17 100 0 100 24 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 6 18 100 0 100 22 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 7 19 100 0 100 21 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 8 20 100 0 100 20 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 9 21 100 0 100 19 0 0 0 0;
0 1 11 1 334 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 10 22 100 0 100 18 0 0 0 0];

%R2L (20 з'єднань)
Warezmaster = [0 1 11 1 36 197 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 100 0 0 255 1 0 5 0 0
39 0 5 0;
156 1 11 1 950 2551 0 0 0 18 0 1 0 0 0 0 21 0 0 0 0 1 1 1 0 0 0 0 100 0 0 218 1 0 3 0 0 1 0 7 0;
0 1 11 1 0 7181 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 1 1 100 0 100 0 0 0 0 0;
0 1 11 1 0 848 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 100 0 0 2 2 100 0 100 0 0 0 0 0;
9 1 11 1 0 5153460 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 100 0 0 3 3 100 0 100 0 0 0 0 0;
10 1 11 1 0 5155468 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 4 4 100 0 100 0 0 0 0 0;
10 1 11 1 0 5151385 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 5 5 100 0 100 0 0 0 0 0;
9 1 11 1 0 5150836 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 6 6 100 0 100 0 0 0 0 0;
10 1 11 1 0 5150823 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 7 7 100 0 100 0 0 0 0 0;
10 1 11 1 0 5150877 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 8 8 100 0 100 0 0 0 0 0;
9 1 11 1 0 5150772 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 9 9 100 0 100 0 0 0 0 0;
10 1 11 1 0 5150841 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 10 10 100 0 100 0 0 0 0
0;
10 1 11 1 0 5150790 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 11 11 100 0 100 0 0 0 0
0;
9 1 11 1 0 5153771 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 12 12 100 0 100 0 0 0 0 0;
9 1 11 1 0 5151049 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 13 13 100 0 100 0 0 0 0 0;
9 1 11 1 0 5149533 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 14 14 100 0 100 0 0 0 0 0;
```

```
9 1 11 1 0 5150938 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 15 15 100 0 100 0 0 0 0 0;
10 1 11 1 0 5151154 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 16 16 100 0 100 0 0 0 0
0;
10 1 11 1 0 5150180 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 17 17 100 0 100 0 0 0 0
0;
2 1 11 1 0 1159100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 100 0 0 18 18 100 0 100 0 0 0 0
0];
```

Додаток Д

Програмна реалізація нейронної мережі MPL

```
clear all

% Ініціалізація змінних

ANum = 64; % кількість атак для навчання НМ
NNum = 16; % кількість нормальних з'єднань для навчання НМ
mcomp = 12; % кількість головних компонент для аналізу
data = zeros(1,mcomp); % обнулення масиву

BadDetectors = 0; % обнулення лічильника для підрахунку відсотка НМ, що
навчилися
cyclevar = 0; % обнулення змінної циклу


atack_data = importdata('d:\my\work\disser\matlab\attack
type\pca_dos_neptune.mat');
type\pca_r2l_imap.mat');
normal_data = importdata('d:\my\work\disser\matlab\attack
type\pca_normal.mat');


atack_train_data = atack_data(:,1:mcomp);
normal_train_data = normal_data(:,1:mcomp);

AS = size(atack_train_data);
atack_size = AS(1);
AN = size(normal_train_data);
normal_size = AN(1);


for i = 1 : ANum
 Poz = randi(atack_size);
 data = cat(1, data, atack_train_data(Poz:));
end

for i = 1 : NNum
 Poz = randi(normal_size);
 data = cat(1, data, normal_train_data(Poz:));
end

data(1:)= [];

train_data = data';

% створення і навчання нейромережевого детектора

TF = zeros(1,ANum + NNum);
for q = 1 : ANum
 TF(1,q)= 1;
end

Net = newff([ minmax(train_data)],[ 10 1], { 'logsig' 'purelin'})
Net.trainParam.epochs = 10000;
Net.trainParam.goal = 0.000001;
[ Net,inf] = train(Net,train_data,TF);
a = sim(Net,train_data);
```

```
% провірка функціонування НМ на невідомих даних

atack = 0;
norm = 0;

data = importdata('d:\my\work\disser\matlab\attack
type\pca_dos_teardrop.mat');
c_data = data(:,1:20);
check_data = c_data';

AS = size(check_data);
atack_size = AS(2)

a = sim(Net,check_data);
for it = 1 : atack_size
 if a(it)> 0.5
 atack = atack + a(1,it);
 end

end

atack
result = (atack / atack_size) * 100
```

Додаток Е

Програмна реалізація нейронної мережі RBF

```
clear all

% Ініціалізація змінних

ANum = 64; % кількість атак для навчання НМ
NNum = 16; % кількість нормальних з'єднань для навчання НМ
mcomp = 12; % кількість головних компонент для аналізу
data = zeros(1,mcomp); % обнулення масиву

BadDetectors = 0; % обнулення лічильника для підрахунку відсотку НМ, що
навчилися
cyclevar = 0; % обнулення змінної циклу


atack_data = importdata('d:\my\work\disser\matlab\attack
type\pca_dos_neptune.mat');
normal_data = importdata('d:\my\work\disser\matlab\attack
type\pca_normal.mat');

atack_train_data = atack_data(:,1:mcomp);
normal_train_data = normal_data(:,1:mcomp);

AS = size(atack_train_data);
atack_size = AS(1);
AN = size(normal_train_data);
normal_size = AN(1);


for i = 1 : ANum
 Poz = randi(atack_size);
 data = cat(1, data, atack_train_data(Poz:));
end

for i = 1 : NNum
 Poz = randi(normal_size);
 data = cat(1, data, normal_train_data(Poz:));
end

data(1:)= [];

train_data = data';

% створення і навчання нейромережевого детектора

TF = zeros(2,ANum + NNum);
for q = 1 : ANum
 TF(1,q)= 1;
end
for q = ANum + 1 : NNum + ANum
 TF(2,q)= 1;
end

[Net,tr] = newrb(train_data, TF);
a = sim(Net,train_data);

% провірка функціонування НМ на невідомих даних
```

```
atack = 0;
norm = 0;

data = importdata('d:\my\work\disser\matlab\attack type\pca_normal.mat');
c_data = data(:,1:20);
check_data = c_data';

AS = size(check_data);
atack_size = AS(2)

a = sim(Net,check_data);
for it = 1 : atack_size
 atack = atack + a(1,it);
 norm = norm + a(2,it);
end

atack
norm
result = (atack / atack_size) * 100
```

Додаток Ж

Програмна реалізація нейронної мережі LVQ

```
clear all

% Ініціалізація змінних

ANum = 64; % кількість атак для навчання НМ
NNum = 16; % кількість нормальних з'єднань для навчання НМ
mcomp = 12; % кількість головних компонент для аналізу
data = zeros(1,mcomp); % обнулення масиву

BadDetectors = 0; % обнулення лічильника для підрахунку відсотку НМ, що
навчилися
cyclevar = 0; % обнулення змінної циклу


atack_data = importdata('d:\my\work\disser\matlab\attack
type\pca_dos_neptune.mat');
normal_data = importdata('d:\my\work\disser\matlab\attack
type\pca_normal.mat');

atack_train_data = atack_data(:,1:mcomp);
normal_train_data = normal_data(:,1:mcomp);

AS = size(atack_train_data);
atack_size = AS(1);
AN = size(normal_train_data);
normal_size = AN(1);


for i = 1 : ANum
 Poz = randi(atack_size);
 data = cat(1, data, atack_train_data(Poz:));
end

for i = 1 : NNum
 Poz = randi(normal_size);
 data = cat(1, data, normal_train_data(Poz:));
end

data(1:)= [];

train_data = data';

% створення і навчання нейромережевого детектора

TF = zeros(2,ANum + NNum);
for q = 1 : ANum
 TF(1,q)= 1;
end
for q = ANum + 1 : NNum + ANum
 TF(2,q)= 1;
end

Net = newlvq(minmax(train_data),10[ .8 .2]);
Net.trainParam.epochs = 200;
Net.trainParam.show = 20;
%Net.trainParam.lr=0.05;
[ Net,inf] = train(Net,train_data,TF);
```

```
a = sim(Net,train_data);

% провірка функціонування НМ на невідомих даних

atack = 0;
norm = 0;
%normal_train_data = normal_train_data';

data = importdata('d:\my\work\disser\matlab\attack type\pca_normal.mat');
c_data = data(:,1:20);
check_data = c_data';

AS = size(check_data);
atack_size = AS(2)

a = sim(Net,check_data);
for it = 1 : atack_size
 atack = atack + a(1,it);
 norm = norm + a(2,it);
end

atack
norm
result = (atack / atack_size) * 100
```

Додаток К

Розподіл в тривимірному просторі мережевих атак і нормальних з’єднань

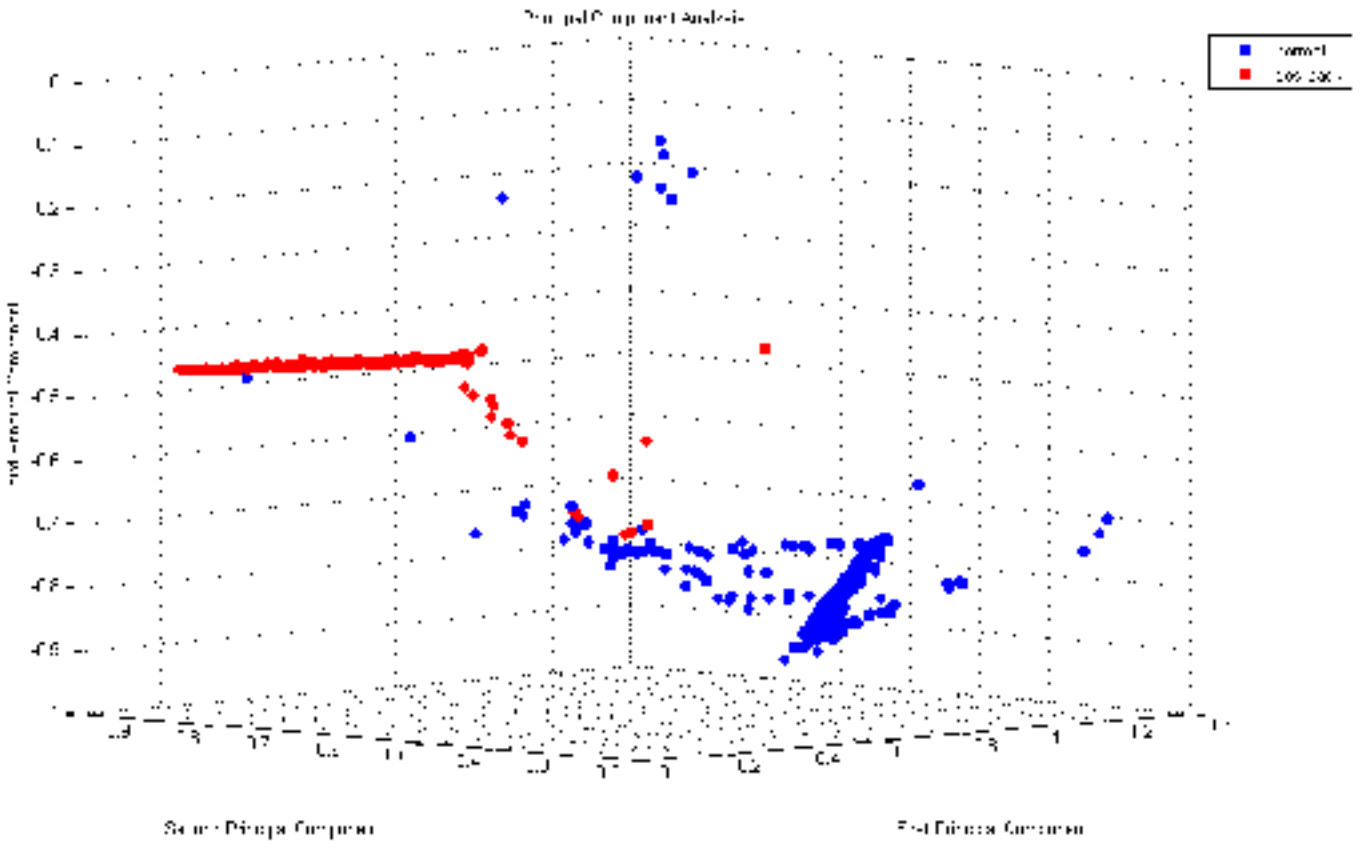


Рисунок К.1 – Розподіл мережевої атаки dos_back і нормальних з’єднань

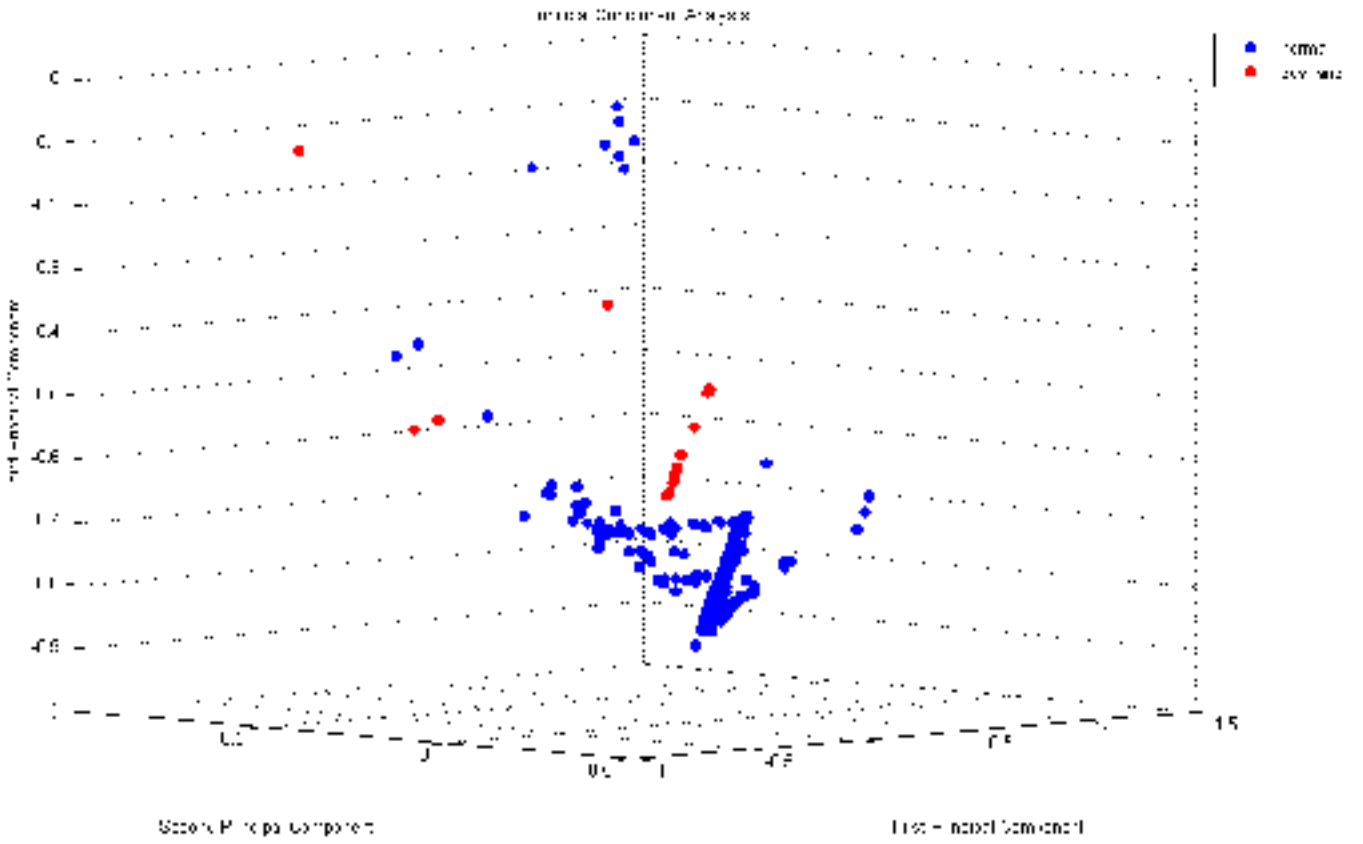


Рисунок К.2 – Розподіл мережевої атаки dos_land і нормальних з’єднань

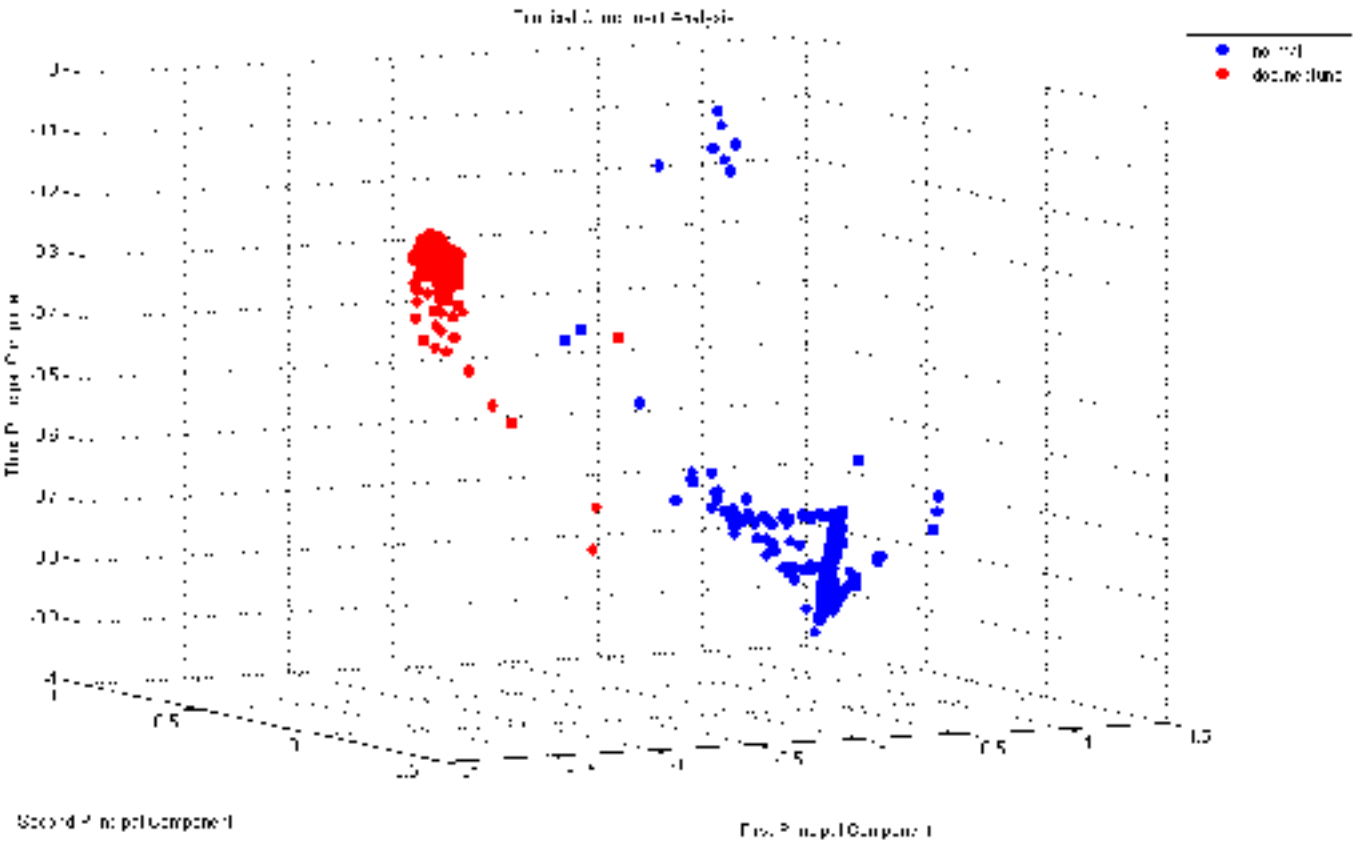


Рисунок К.3 – Розподіл мережевої атаки dos_back і нормальних з’єднань

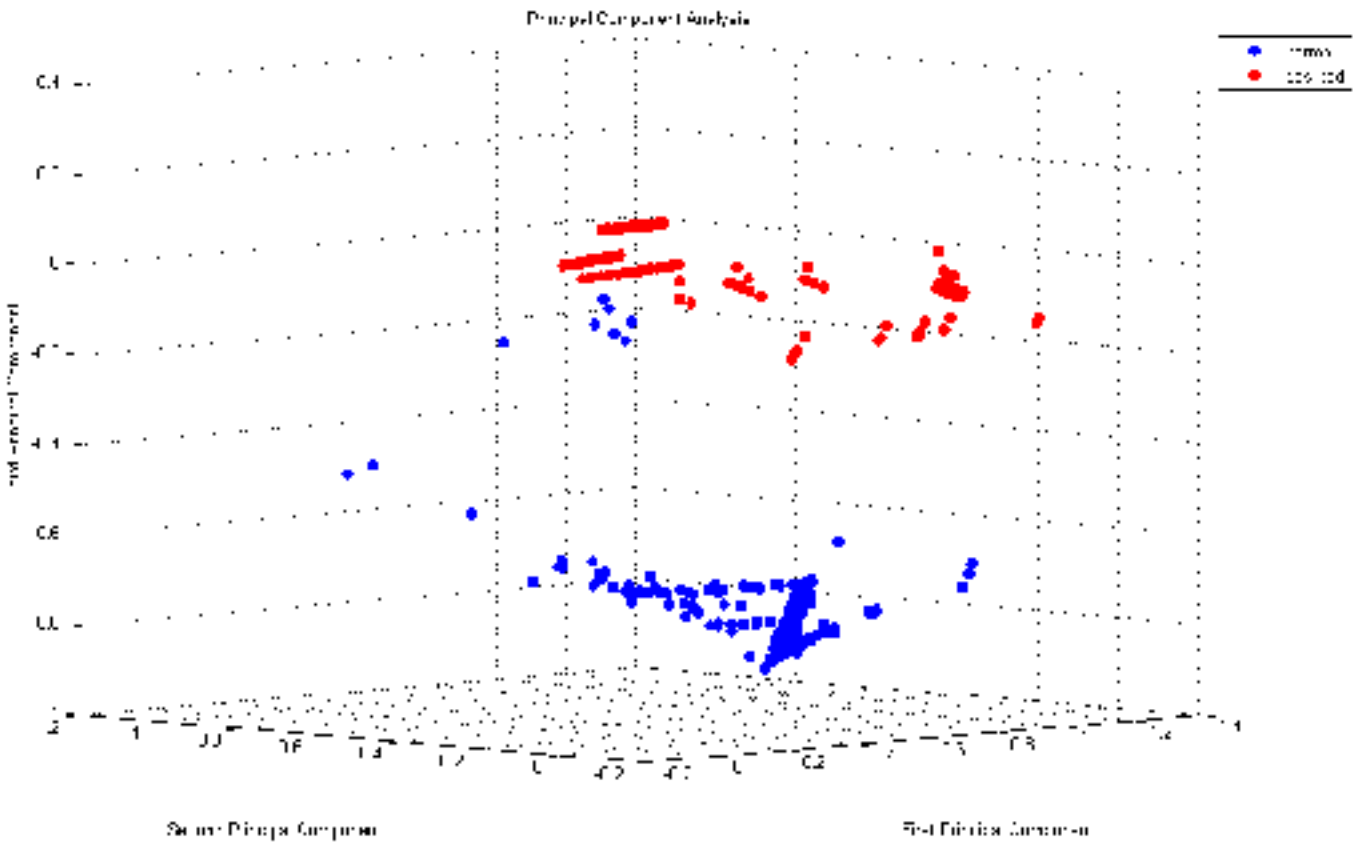


Рисунок К.4 – Розподіл мережевої атаки dos_pod і нормальних з’єднань

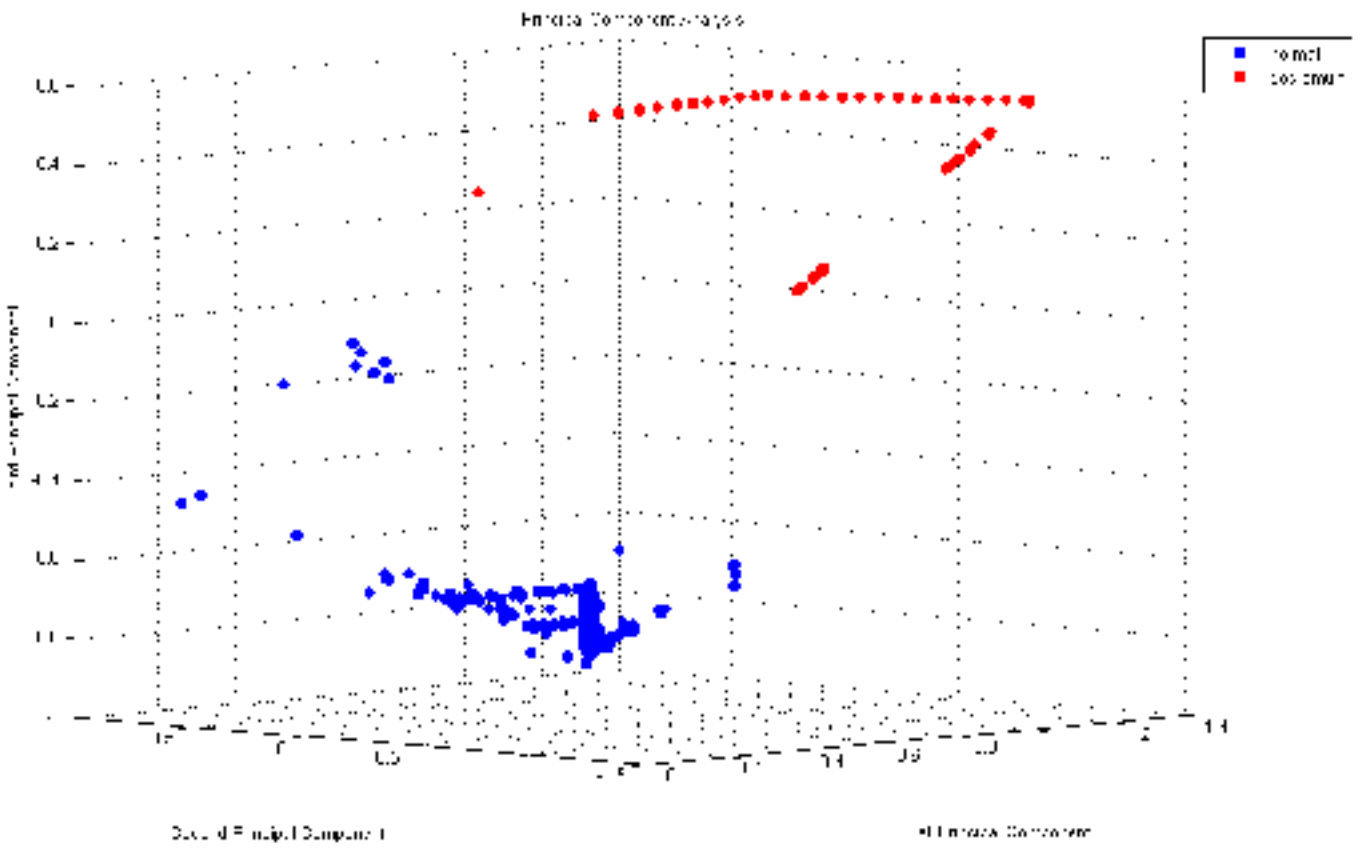


Рисунок К.5 – Розподіл мережевої атаки dos_smurf і нормальних з’єднань

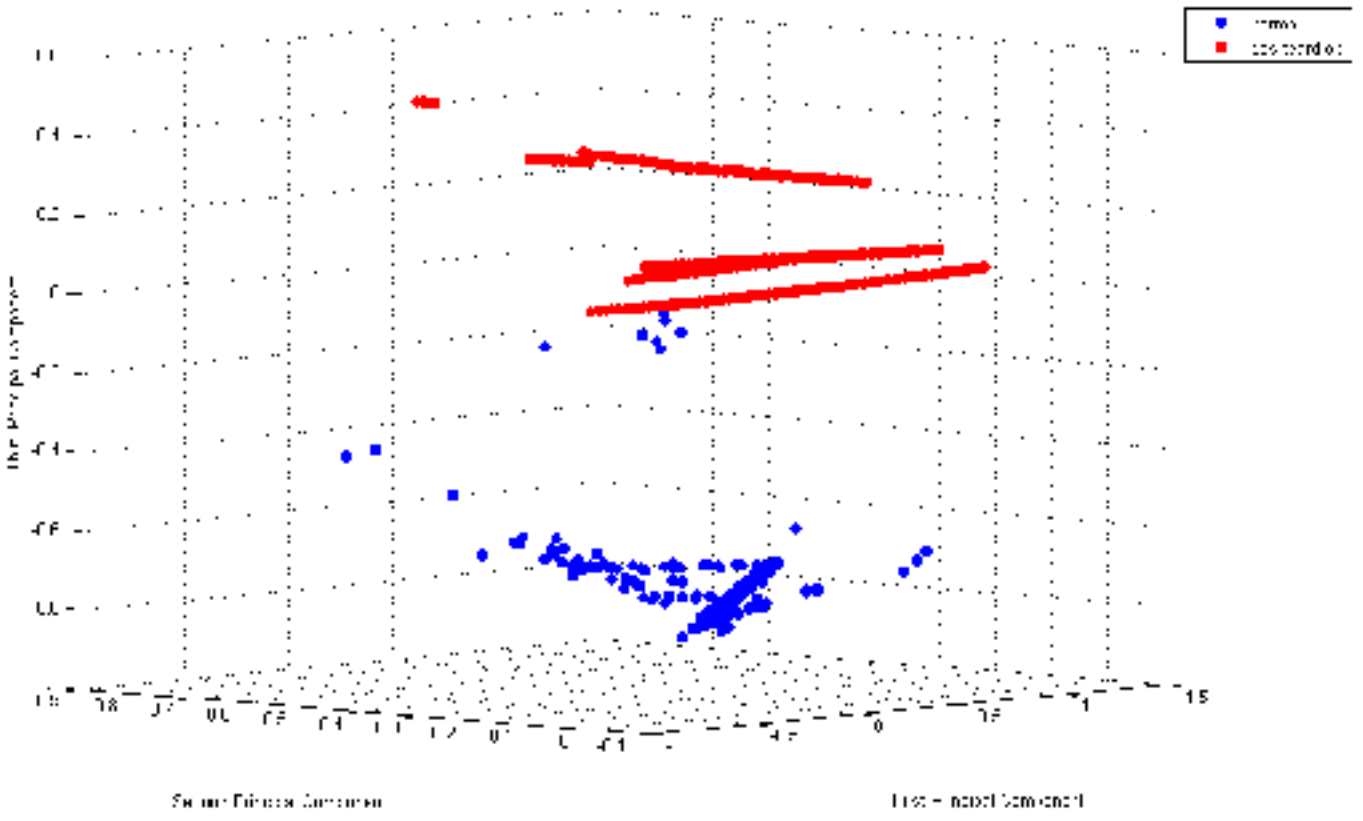


Рисунок К.6 – Розподіл мережевої атаки dos_teardrop і нормальних з’єднань

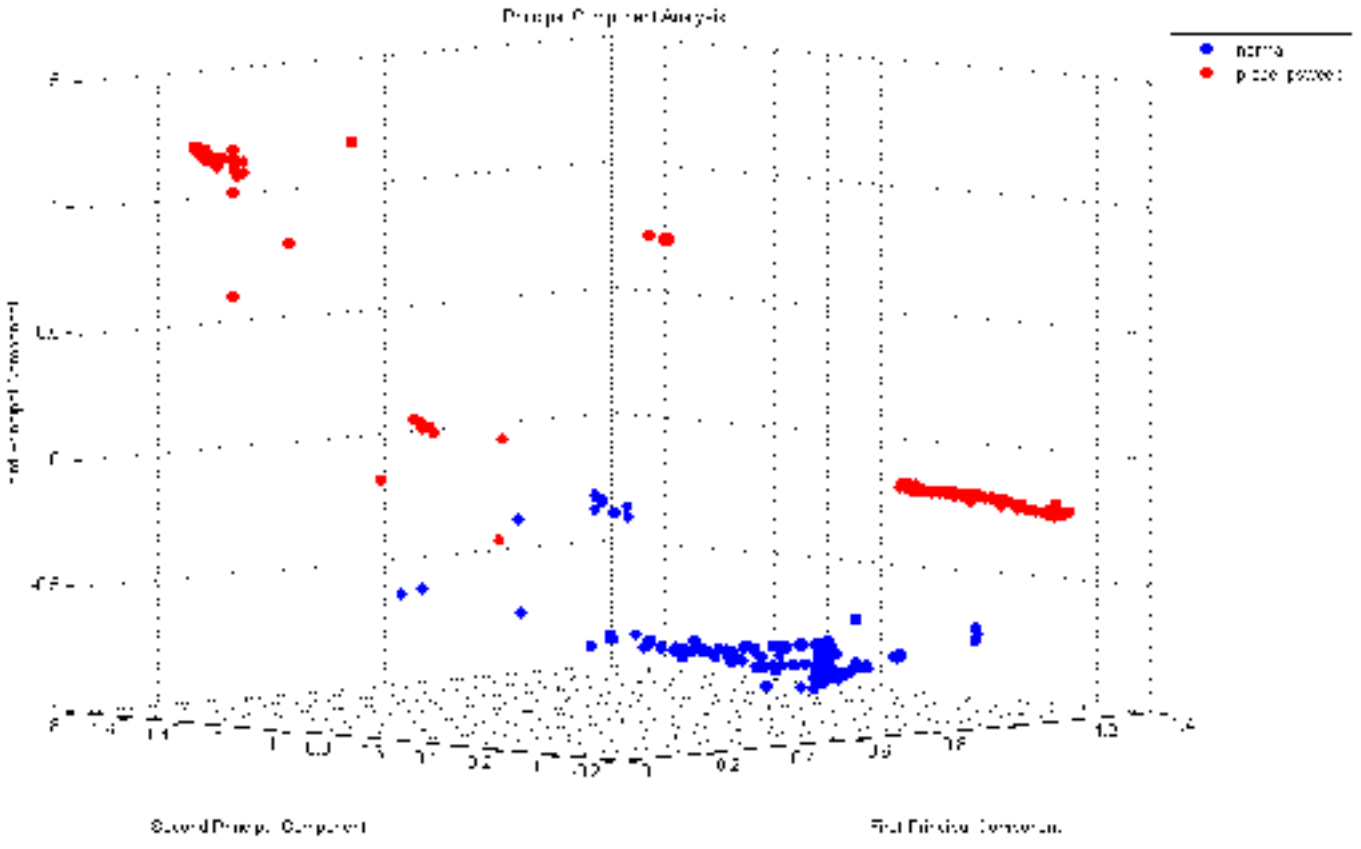


Рисунок К.7 – Розподіл мережевої атаки probe_ipsweep і нормальних з’єднань

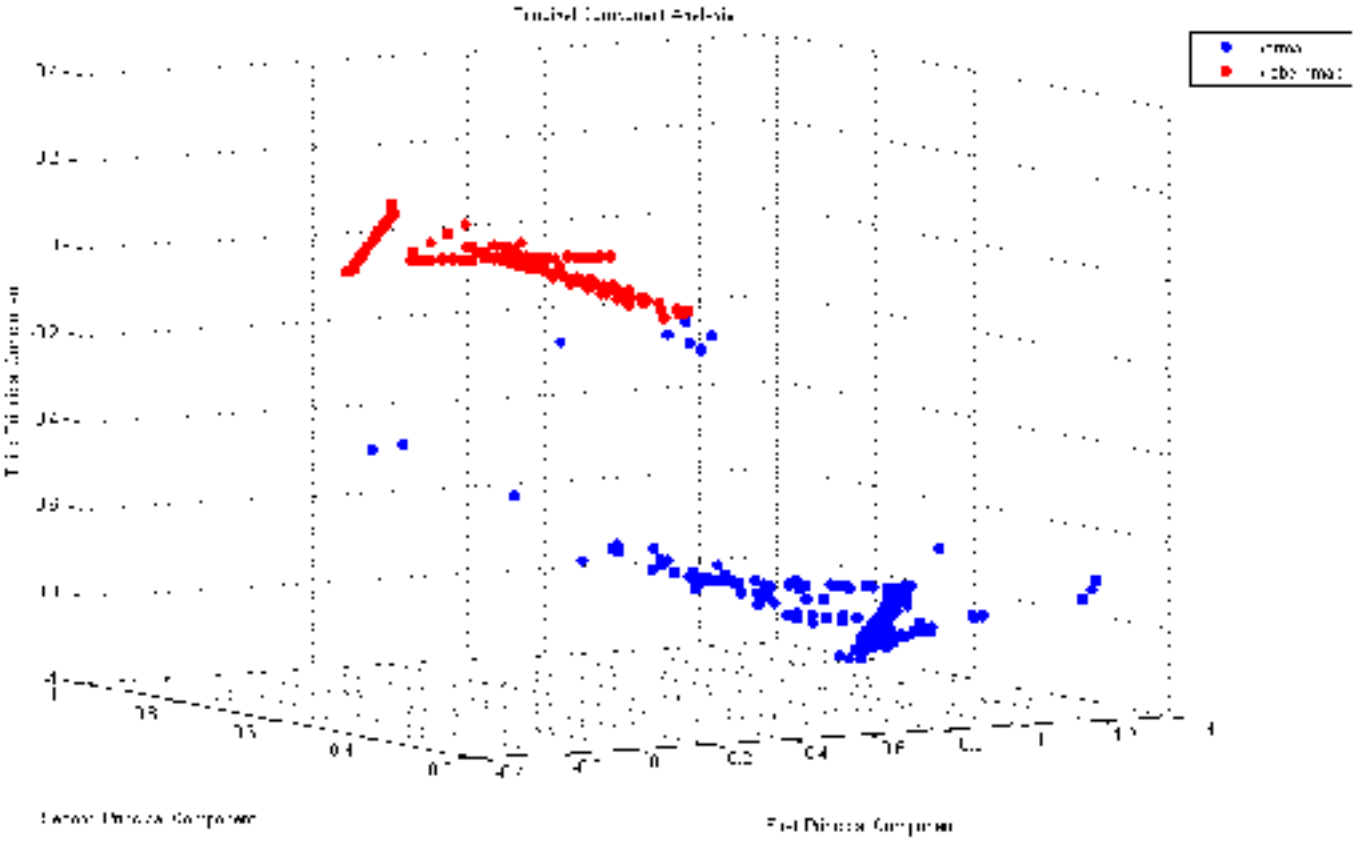


Рисунок К.8 – Розподіл мережевої атаки probe_nmap і нормальних з’єднань

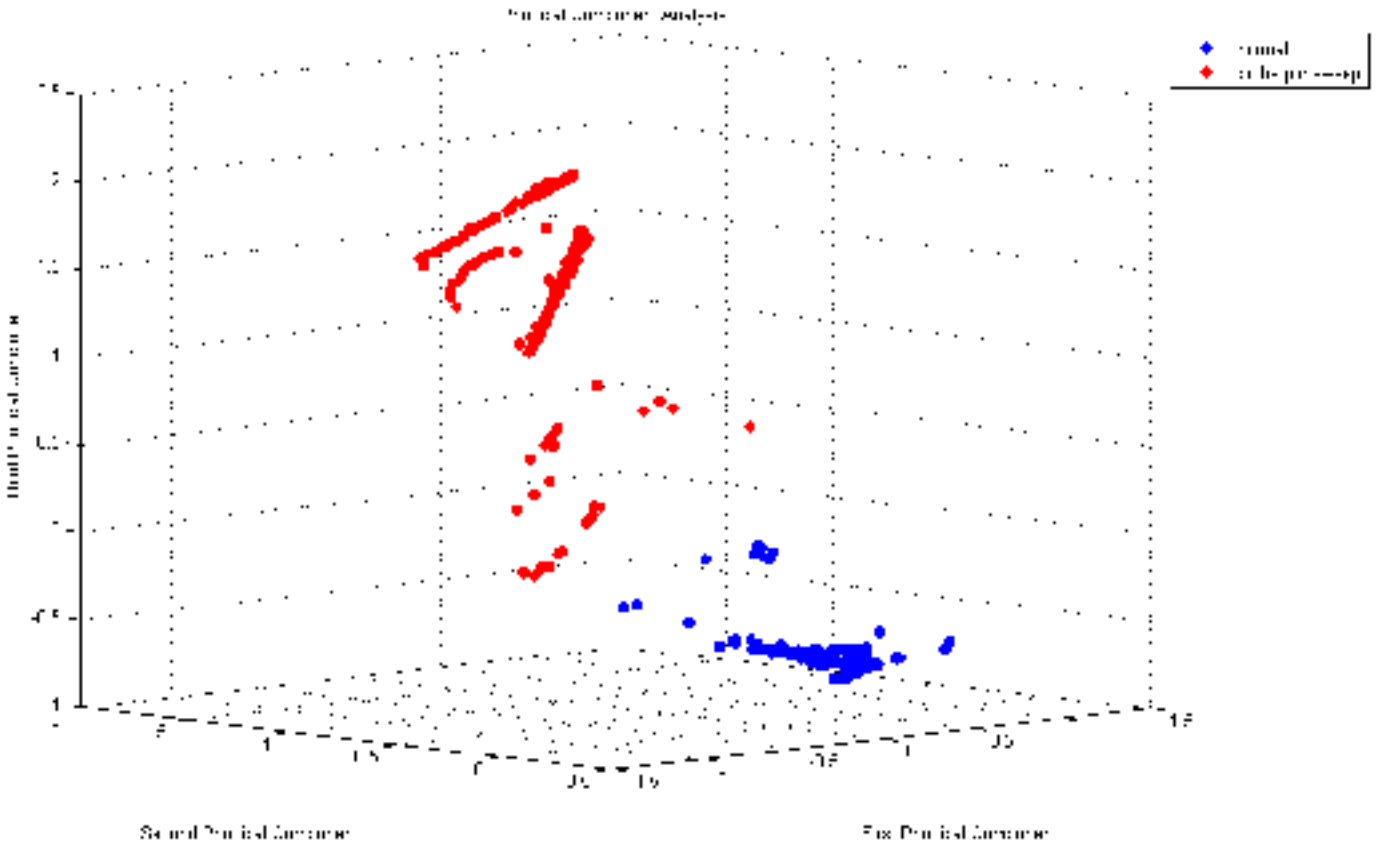


Рисунок К.9 – Розподіл мережевої атаки probe_portsweep і нормальних з’єднань

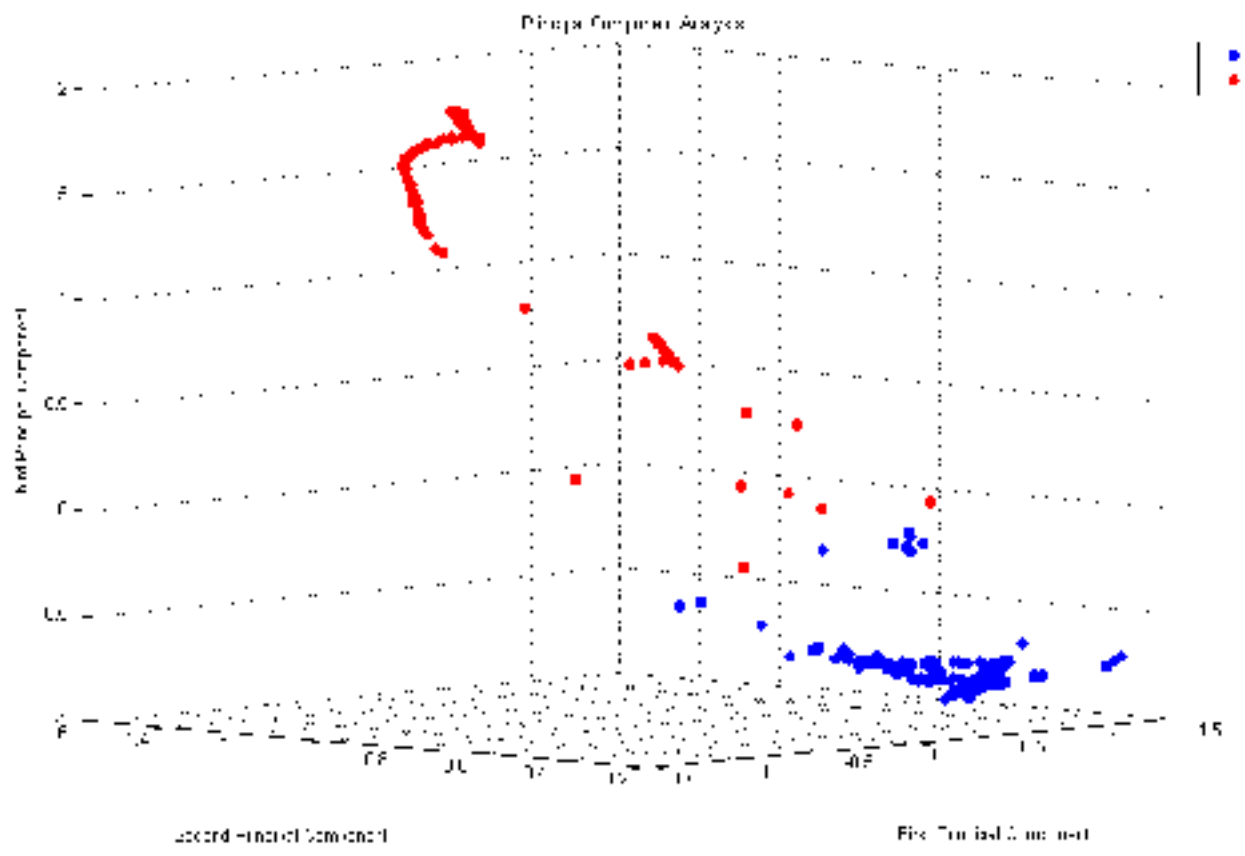


Рисунок К.10 – Розподіл мережевої атаки probe_satan і нормальних з’єднань

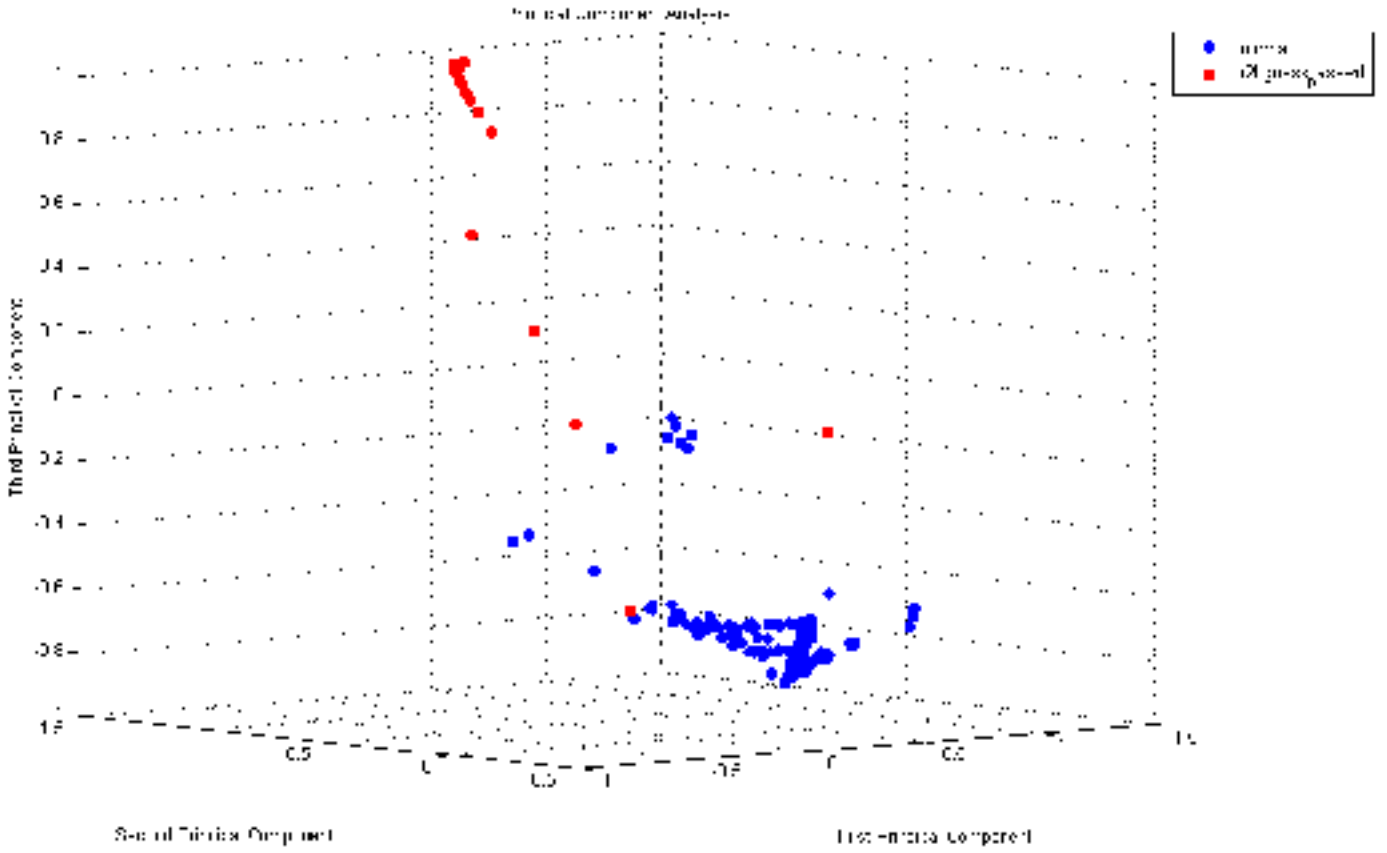


Рисунок К.11 – Розподіл мережевої атаки r2l_guess_passwd і нормальних з’єднань

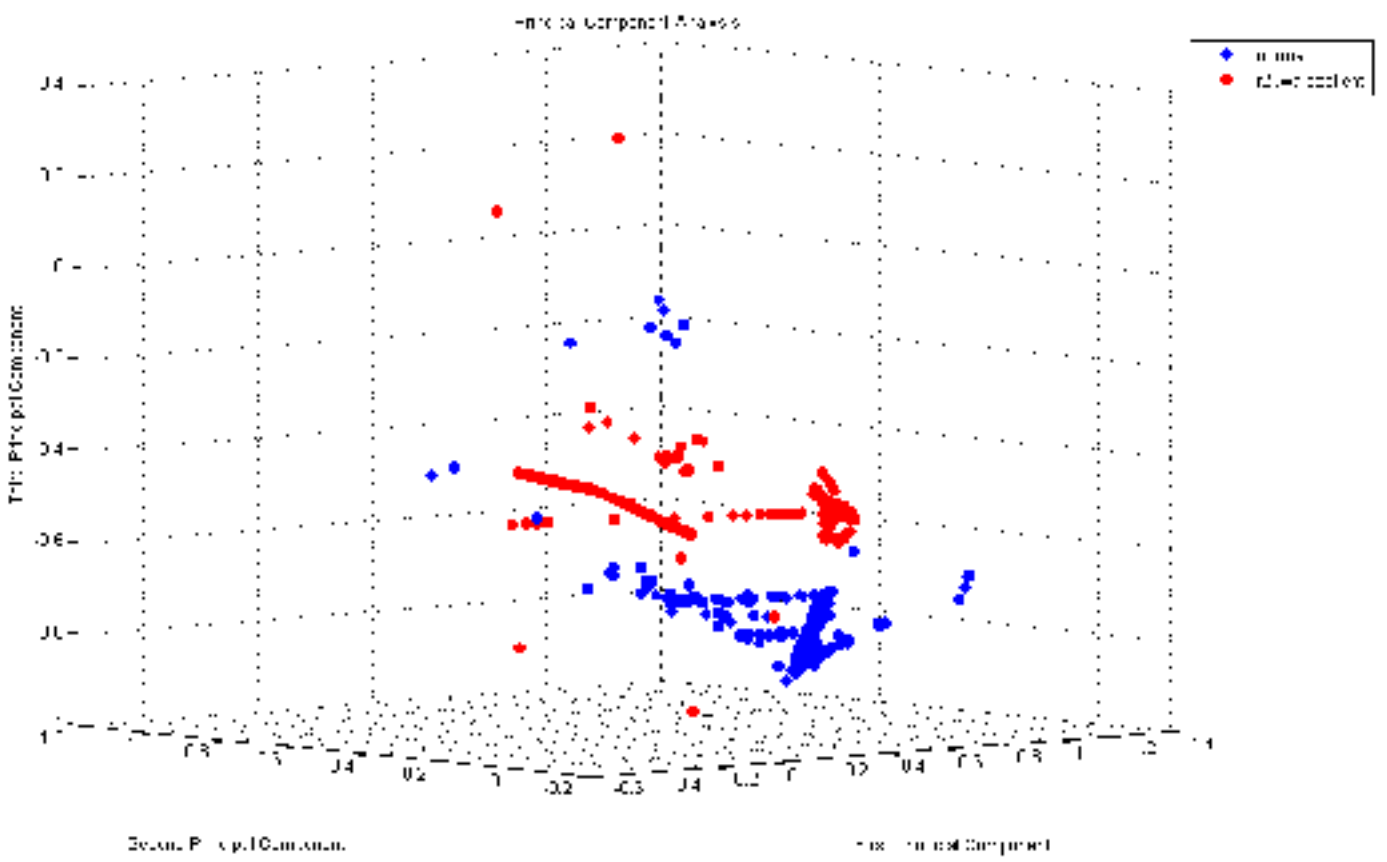


Рисунок К.12 – Розподіл мережевої атаки r2l_warezclient і нормальних з’єднань

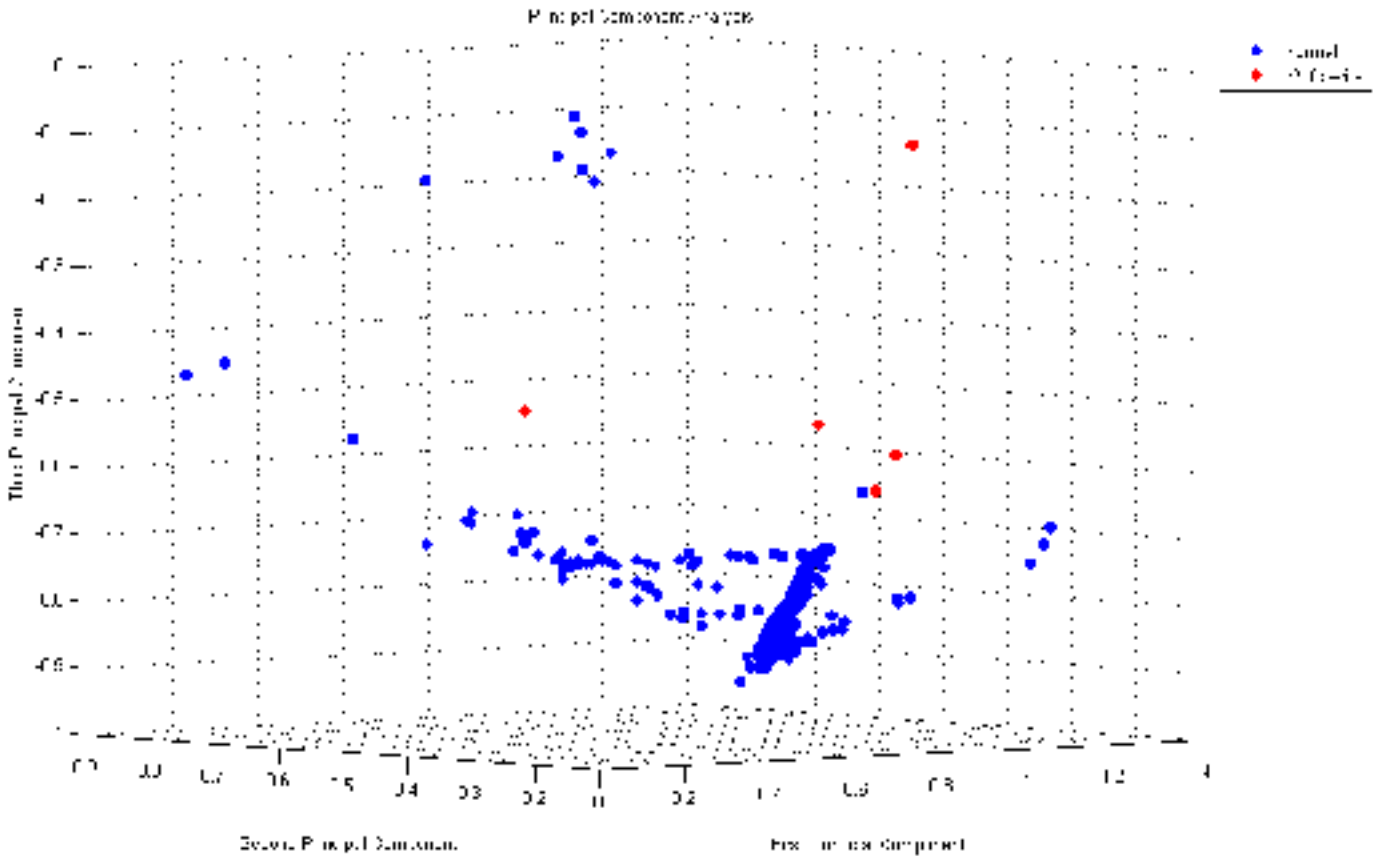


Рисунок К.13 – Розподіл мережевої атаки r2l_ftpwrite і нормальних з’єднань

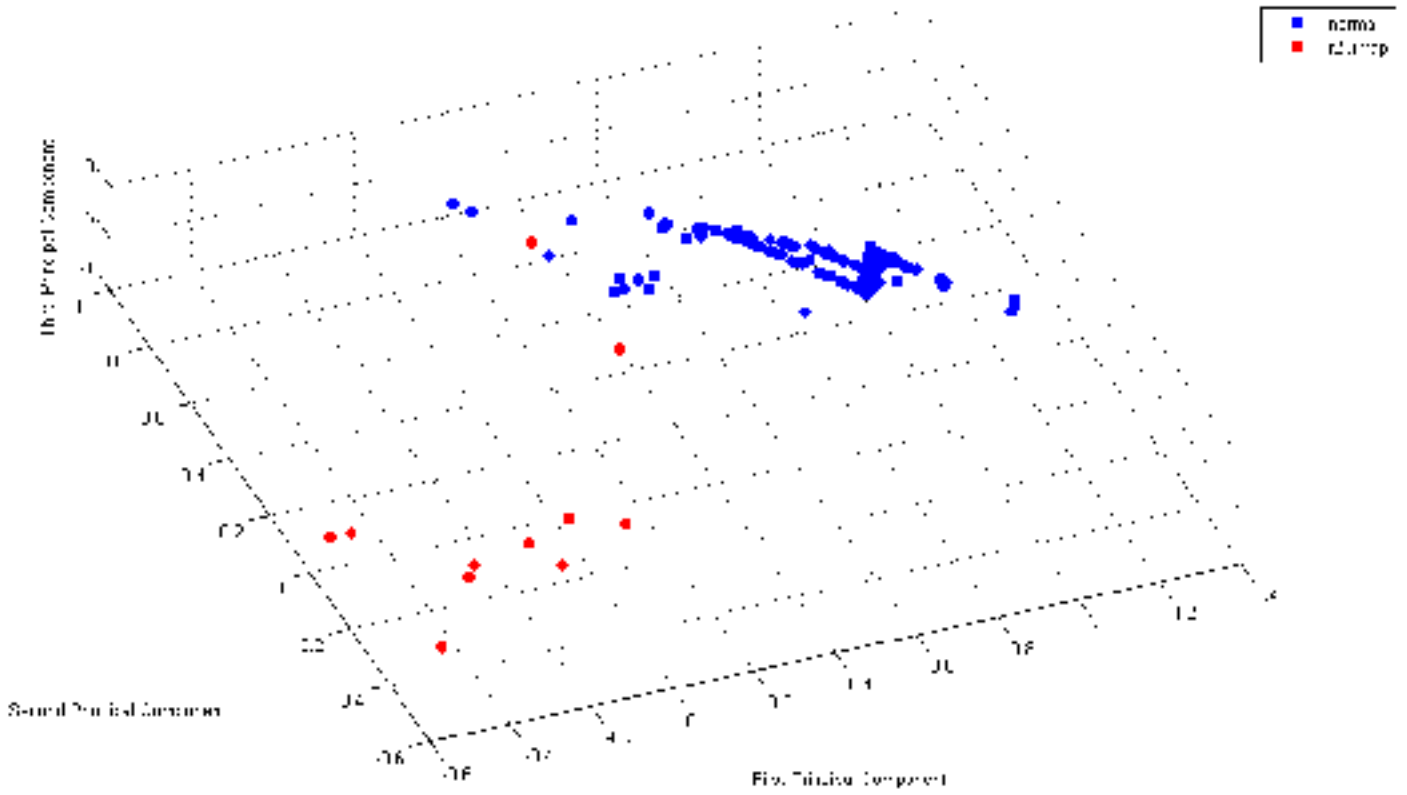


Рисунок К.14 – Розподіл мережевої атаки r2l_imap і нормальних з’єднань

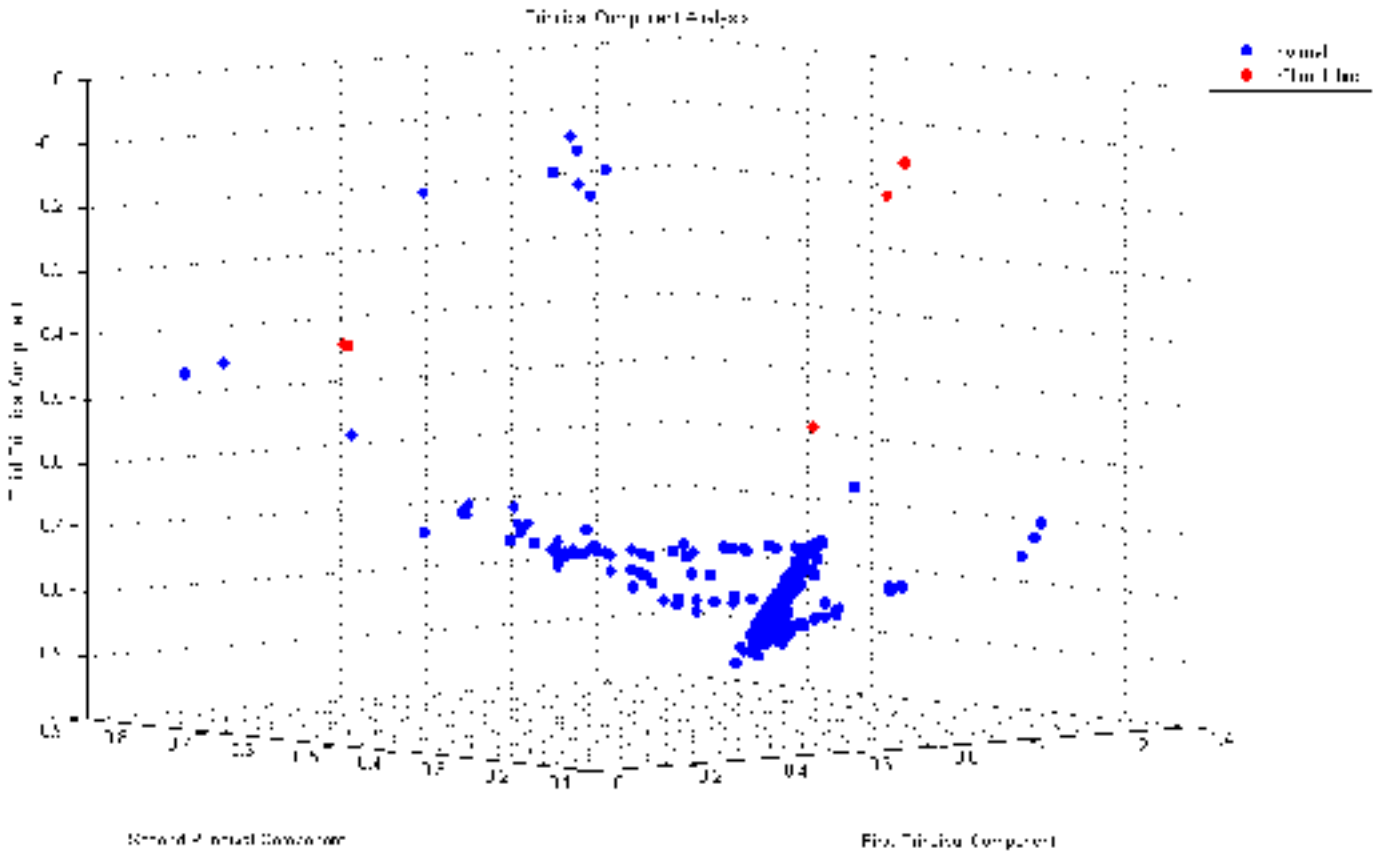


Рисунок К.15 – Розподіл мережевої атаки r2l_multihop і нормальних з’єднань

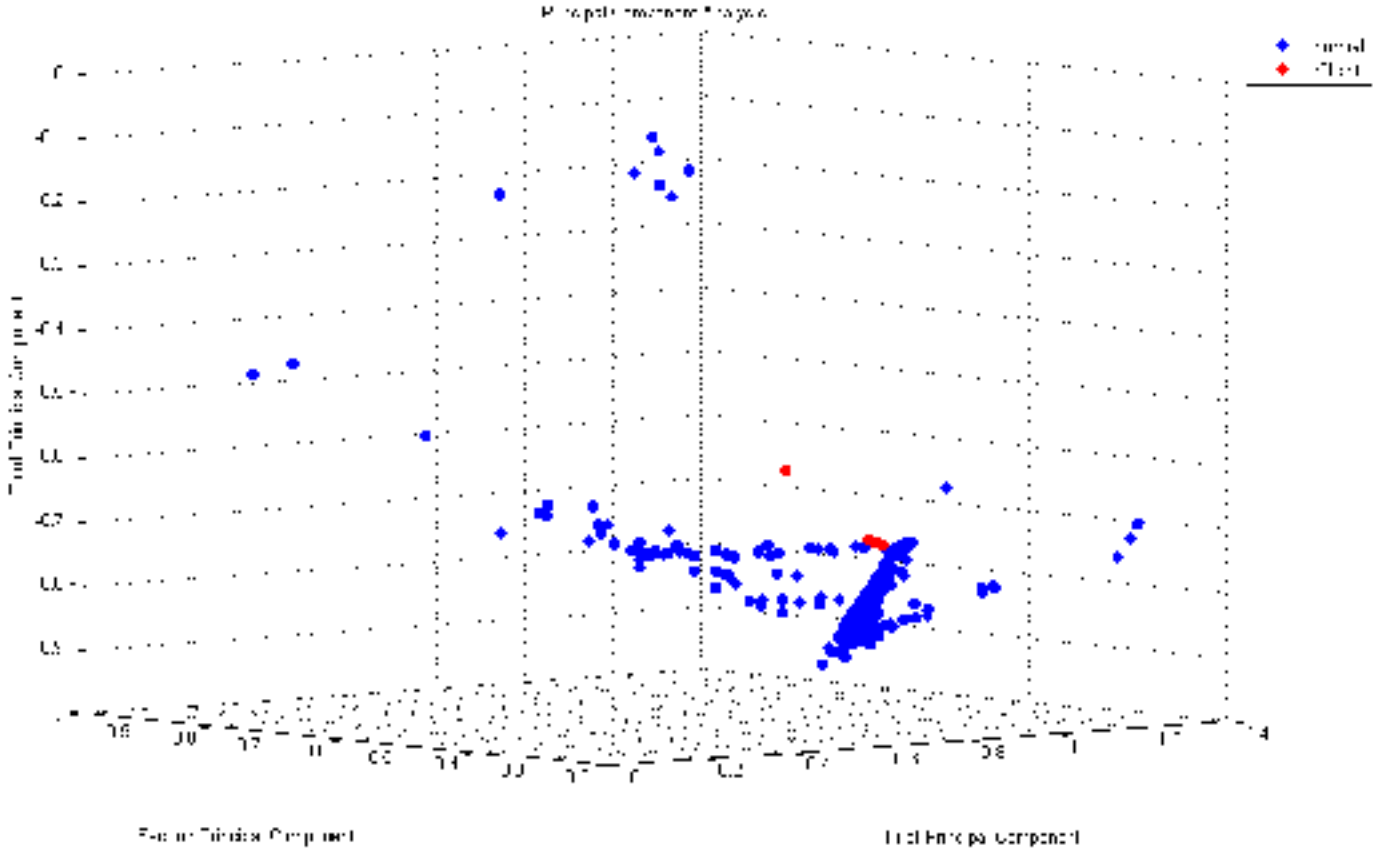


Рисунок К.16 – Розподіл мережевої атаки r2l_phf і нормальних з’єднань

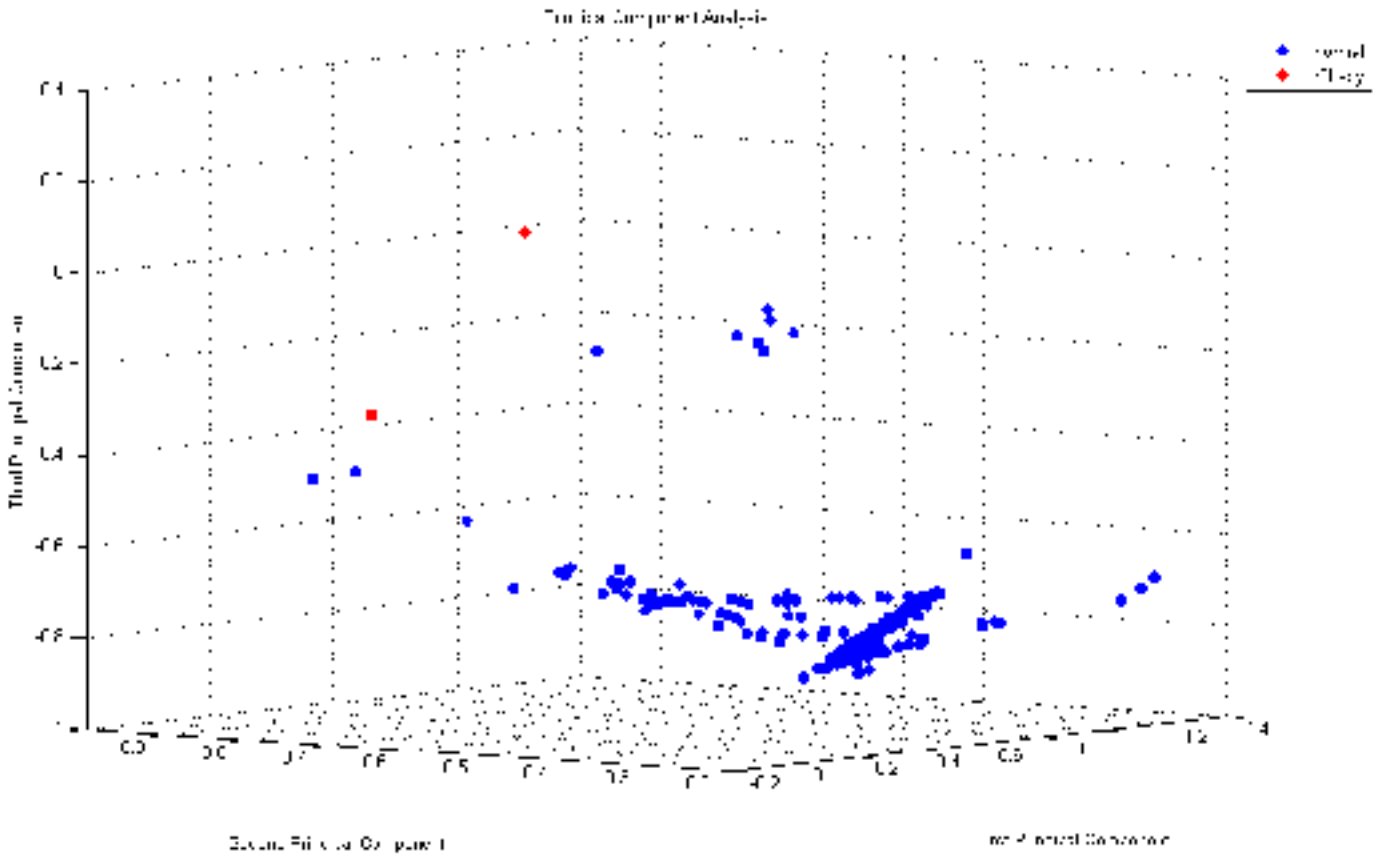


Рисунок К.17 – Розподіл мережевої атаки r2l_spy і нормальних з’єднань

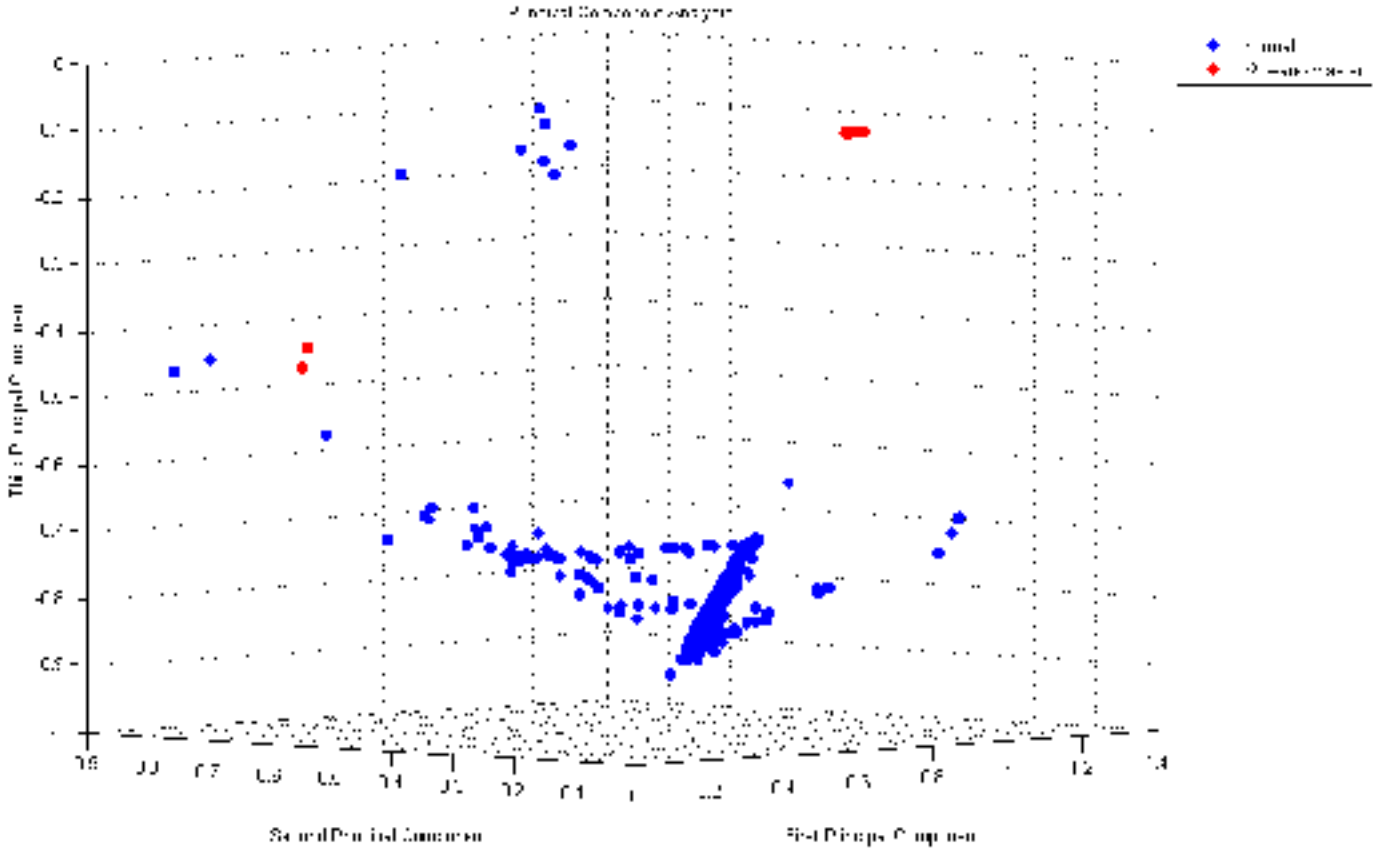


Рисунок К.18 – Розподіл мережевої атаки r2l_warezmaster і нормальних з’єднань

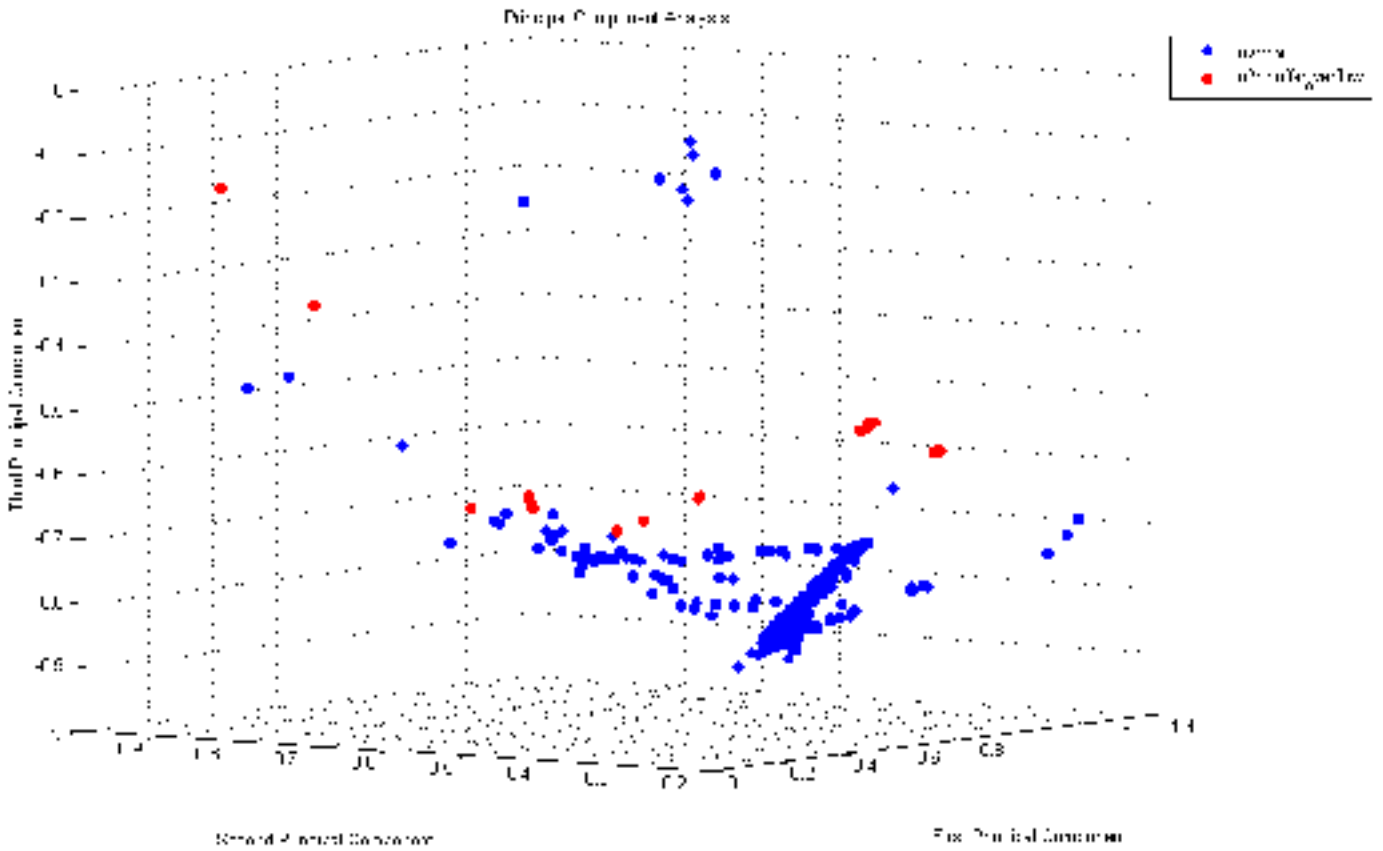


Рисунок К.19 – Розподіл мережевої атаки u2r_bufferoverflow і нормальних з’єднань

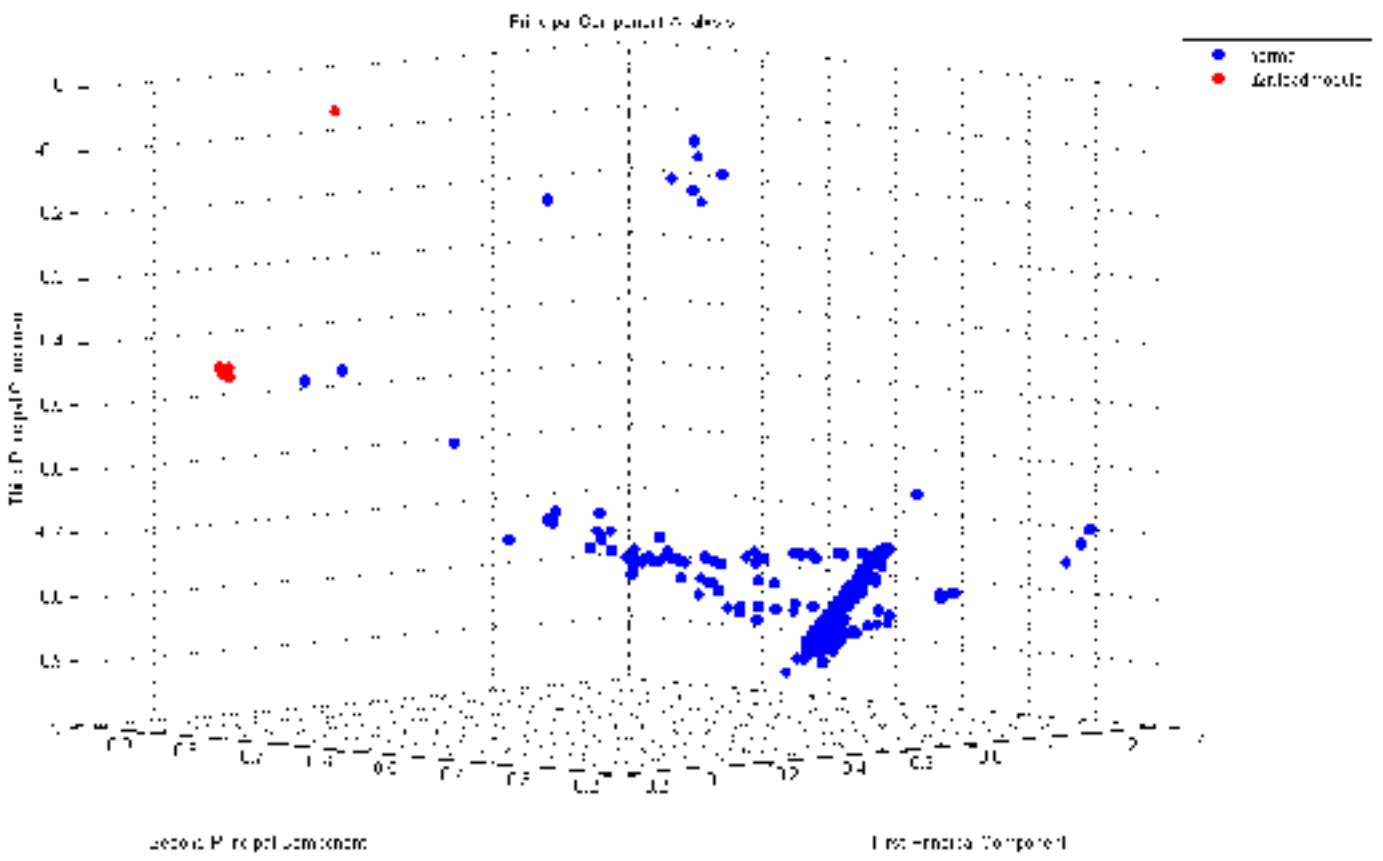


Рисунок К.20 – Розподіл мережевої атаки u2r_loadmodule і нормальних з’єднань

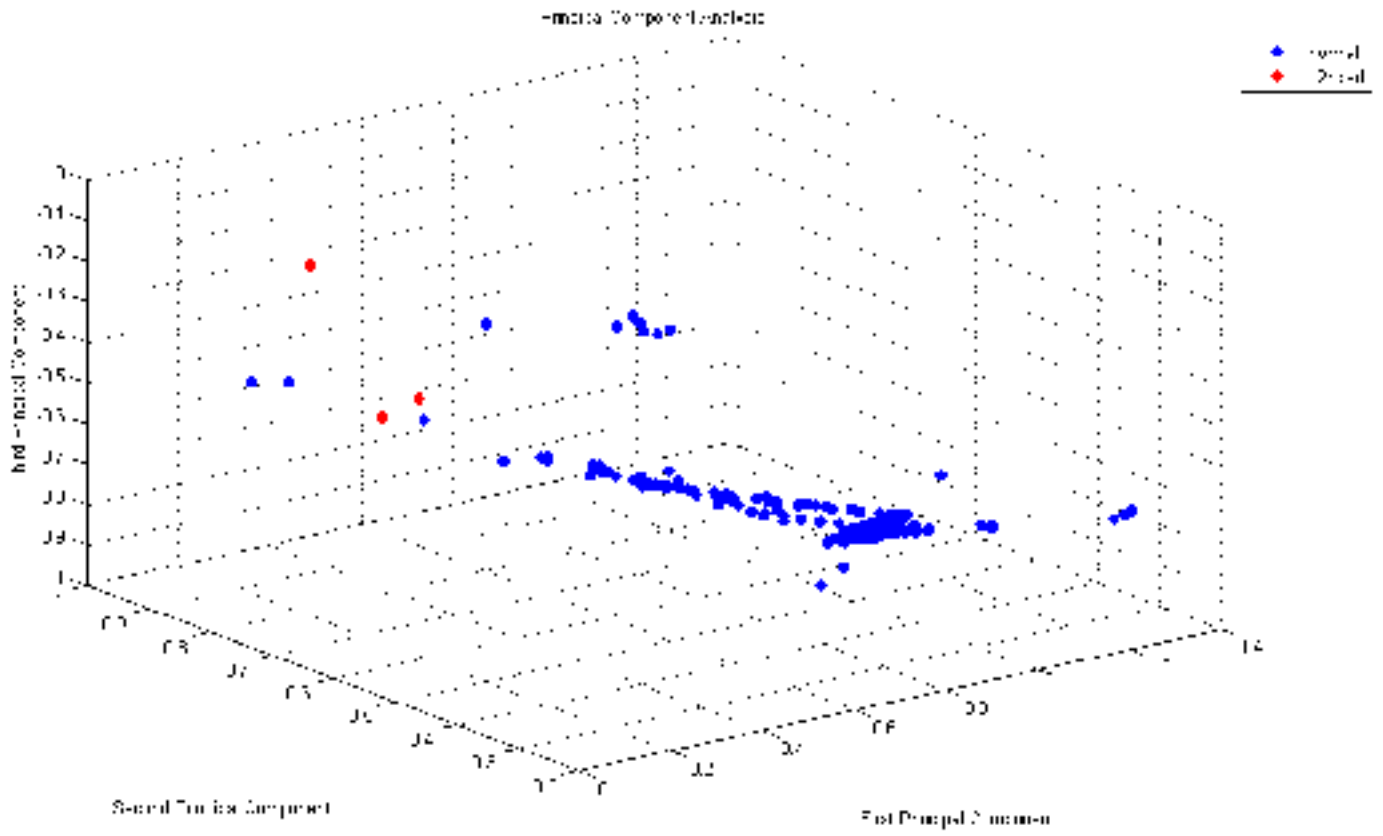


Рисунок К.21 – Розподіл мережевої атаки u2r_perl і нормальних з’єднань

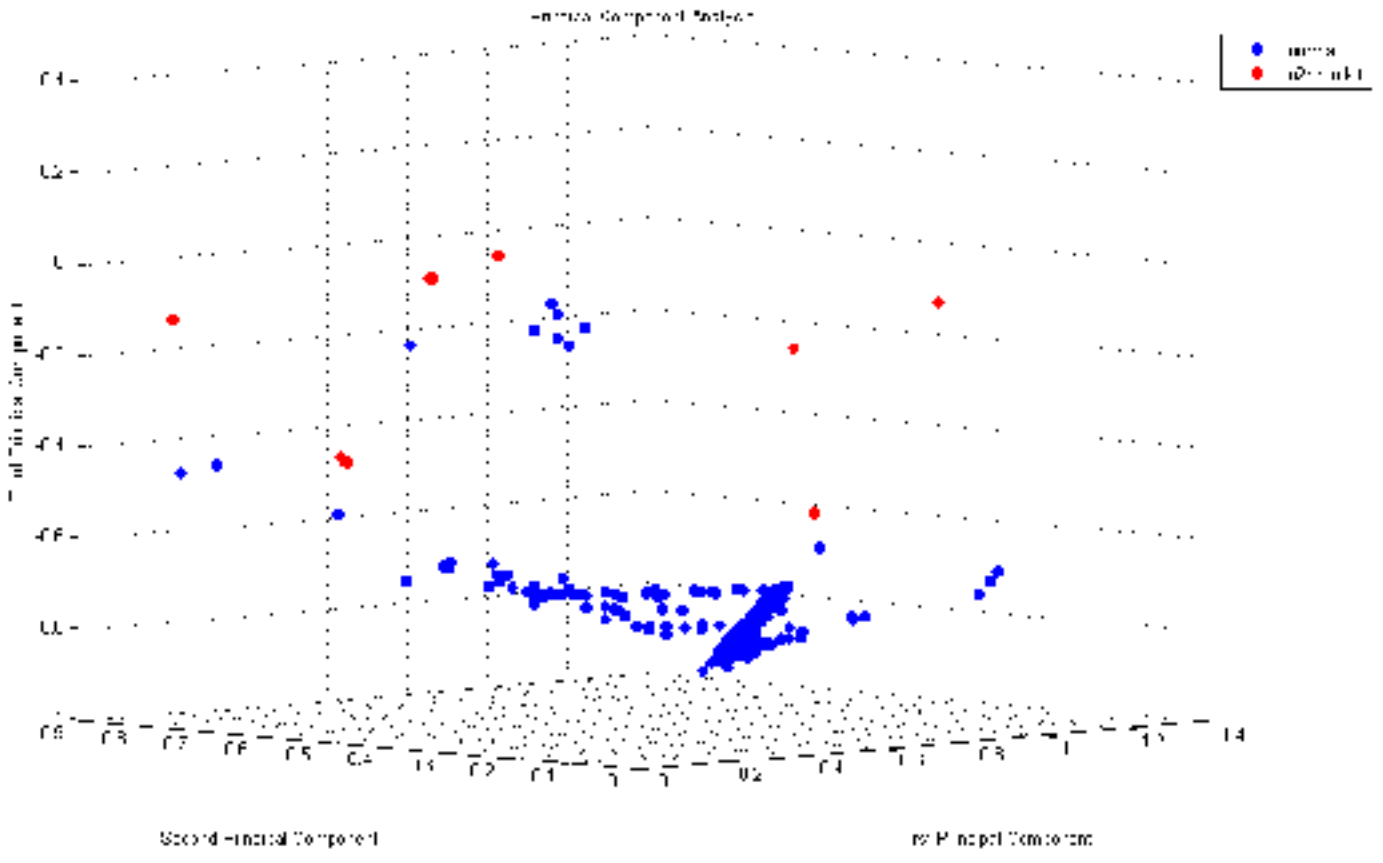


Рисунок К.22 – Розподіл мережевої атаки u2r_rootkit і нормальних з’єднань

Додаток Л

Фрагменти лістингів програмного забезпечення системи виявлення і класифікації атак

```
#include <stdafx.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#include <math.h>
#include <time.h>
#include <stdio.h>
#include <process.h>
#include <string.h>
#include "Filer.h"

//опис параметрів
      const int input = 12; // кількість вхідних нейронів
      const int hid = 10; // кількість нейронів прихованого шару
      const int out = 2; // кількість нейронів вихідного шару
      const int FileNumbers = 6;
      const int ConnNumbers = 7;
      const int ClearNumber = 5;
      char PathNormal[ 100] = "D:/KDD/Learning/Normal/";
      char PathAttack[ 100] = "D:/KDD/Learning/Attack/";
      char PathCheck[ 100] = "D:/KDD/";

      int minindex; // нейрон-переможець
      int error = 0;
      int nnin[ input] ; // масив вхідних елементів
      double nnhid[ hid] ; // масив прихованих елементів
      double nnout[ out] ; // масив вихідних елементів
      double Omegaij[ hid][ input] ; // масив вагових коефіцієнтів між вхідним і
прихованим шаром
      int Vjk[ out][ hid] ; // масив вагових коефіцієнтів між прихованим і
вихідним шаром

      double rand01;
      int rand02;
      double rand1;
      FILE *stream;
      FILE *stream1;

int _calculationHID()
{
      int index;
// обчислення значень прихованих елементів
      for (int i = 0; i < hid; i++)
      {
      double temp = 0;
      for (int j = 0; j < input; j++)
      temp = temp + ((nnin[ j] -Omegaij[ i][ j] )*(nnin[ j] -Omegaij[ i][ j] )); //
формула обчислення вагових коефіцієнтів між вхідним і прихованим шаром
      nnhid[ i] = sqrt(temp);
      }

// визначення нейрона-переможця
      index = 0;
      for (int i = 1; i < hid; i++)
```

```
        {
        if (nnhid[ i] < nnhid[ index] )
        index = i;
        }
        return index;
}

void _calculationLVQ ()
{

        minindex = _calculationHID();
        for (int i = 0; i < hid; i++)
        nnhid[ i] = 0;
        nnhid[ minindex] = 1;

// обчислення значень вихідних елементів
        for (int i = 0; i < out; i++)
        {
        double temp = 0;
        for (int j = 0; j < hid; j++)
        temp = temp + (nnhid[ j] * Vjk[ i][ j] );
        nnout[ i] = temp;
        }
}

double K(double ro)
{
        ro = -0.01*ro;
        return exp(ro);
}
```

```
void _learning ()
{
// архітектура нейронної мережі для побудови детектора
// nnin - вхідний шар, nnhid - прихований шар, nnout - вихідний шар

int all = 0;
for (;all == 0;)
{

        for (int до = 0; до < (hid*ClearNumber/FileNumbers); k++)
        {
        Vjk[ 0][ k] = 1;
        Vjk[ 1][ k] = 0;
        }
        for (int до = (hid*ClearNumber/FileNumbers); до < hid; k++)
        {
        Vjk[ 0][ k] = 0;
        Vjk[ 1][ k] = 1;
        }

        int FData[ FileNumbers][ ConnNumbers][ input] ; //масив для зберігання
навчальної вибірки
        int Etalon[ FileNumbers][ ConnNumbers][ out] ; //масив еталонних значень
        bool CoolCycle;

        //генерація еталонних значень
        for (int i = 0; i < ClearNumber; i++)
        for (int j = 0; j < ConnNumbers; j++)
        {
        Etalon[ i][ j][ 0] = 1;
```

```
        Etalon[ i][ j][ 1] = 0;
        }
        for (int i = ClearNumber; i < FileNumbers; i++)
        for (int j = 0; j < ConnNumbers; j++)
        {
        Etalon[ i][ j][ 0] = 0;
        Etalon[ i][ j][ 1] = 1;
        }

        //читання даних з файлу
        stBufFile _BufFile; //Filer.h


// формування навчальної вибірки
        struct _finddata_t c_file;
        intptr_t hFile;
        int check = 1;
        fopen_s( &stream1, "d:/datafile1.dat", "wt" );

        for (int i = 0; i < FileNumbers; i++)
        {

        int number = 0;
        if (check <= ClearNumber)
        {
        if( (hFile = _findfirst( "D:/KDD/*.*", &c_file )) == -1L )
        printf( "No *.* files in current directory!\n" );
        else
        {
        do
        {
        number ++;
        }
        while( _findnext( hFile &c_file )== 0 );
        _findclose( hFile );

        int RANGE_MIN = 1;
        int RANGE_MAX = number;
        rand1 = (((double) rand() / (double) RAND_MAX) * RANGE_MAX + RANGE_MIN);
        printf("%f \n", rand1);

        hFile = _findfirst( "D:/KDD/*.*", &c_file );
        for (int p = 1; p < rand1-1; p++)
        {
        _findnext( hFile &c_file );
        }
        }
        printf ("%d", number);

        int fff = 0;
        while (PathClear[ fff] != '\0' )
        {
        PathTemp[ fff] = PathClear[ fff] ;
        fff++;
        }
        PathTemp[ fff] = '\0' ;
        int ff = 0;
        while (c_file.name[ ff] != '\0' )
        ff++;

        fprintf_s( stream1, "%s\n", c_file.name);
```

```
for (int f = 0; f < ff; f++)
{
PathTemp[ fff + f] = c_file.name[ f] ;
PathTemp[ fff + f+1] = '\0' ;
}

_BufFile = ReturnBufFile(PathTemp);
}

else
{
if( (hFile = _findfirst("D:/KDD/*.*", &c_file )) == -1L )
printf( "No *.* files in current directory!\n" );
else
{
do
{
number ++;
}
while( _findnext( hFile &c_file )== 0 );
_findclose( hFile );

int RANGE_MIN = 1;
int RANGE_MAX = number;
rand1 = (((double) rand() / (double) RAND_MAX) * RANGE_MAX + RANGE_MIN);
printf("%f \n", rand1);

hFile = _findfirst( "D:/KDD/*.*", &c_file );
for (int p = 1; p < rand1-1; p++)
{
_findnext( hFile &c_file );
}
}
printf ("%d", number);

int fff = 0;
while (PathVirus[ fff] !='\0' )
{
PathTemp[ fff] = PathVirus[ fff] ;
fff++;
}
PathTemp[ fff] = '\0' ;
int ff = 0;
while (c_file.name[ ff] !='\0' )
ff++;

fprintf_s( stream1, "%s\n", c_file.name);

for (int f = 0; f < ff; f++)
{
PathTemp[ fff + f] = c_file.name[ f] ;
PathTemp[ fff + f+1] = '\0' ;
}

_BufFile = ReturnBufFile(PathTemp);
}

for (int j = 0; j < ConnNumbers; j++)
{
int RANGE_MIN = 1;
int RANGE_MAX = _BufFile.iSize-input;
rand02 = (((double) rand() / (double) RAND_MAX) * RANGE_MAX +
```

```
RANGE_MIN);

      for (int до = 0; до < input; k++)
      FData[ i][ j][ k] = _BufFile.cBuf[ rand02+k] ;
      }
      PathTemp[ 0] = '\0' ;
      check = check + 1;
      _findclose( hFile );

      }

      fclose( stream1 );


int t = 1;
CoolCycle = false;
// ініціалізація масиву
for (int i = 0; i < hid; i++)
      for (int j = 0; j < input; j++)
      {
      rand01 =((double) rand() / (double)RAND_MAX)*255;
      Omegaij[ i][ j] = rand01;
      }

for (; (!CoolCycle && t < 50);)
{
      for (int OutCycle = 0; (!CoolCycle && (OutCycle < FileNumbers));
OutCycle++)
      {
      for (int InCycle = 0; (!CoolCycle && (InCycle < ConnNumbers));
InCycle++)
      {
      // передаємо дані мережевого з'єднання на вхід нейронної мережі
      for (int i = 0; i < input; i++)
      nnin[ i] = FData[ OutCycle][ InCycle][ i] ;


      // аналіз
      minindex = _calculationHID ();

// відображення даних
      printf("\n Normal %1f %1d\n",nnout[ 0] ,Etalon[ OutCycle][ InCycle][ 0] );
      printf(" Attack %1f %1d\n",nnout[ 1] ,Etalon[ OutCycle][ InCycle][ 1] );

      //навчання нейронної мережі
      if (nnout[ 0] == Etalon[ OutCycle][ InCycle][ 0] )
      {

      for (int i = 0; i < hid; i++)
      {
      int a = abs(Vjk[ 0][ minindex] - Etalon[ OutCycle][ InCycle][ 0] );
      for (int j = 0; j < input; j++)
      {
      printf("\n do %1f",Omegaij[ i][ j] );
      Omegaij[ i][ j] = Omegaij[ i][ j] + ((double) (1-2*a) / (double) t) * (nnin[ j]
- Omegaij[ i][ j] )*K(nnhid[ i] );
      Omegaij[ minindex][ j] = Omegaij[ minindex][ j] + ((double) (1-2*a) / (double)
t) * (nnin[ j] - Omegaij[ minindex][ j] );
      printf(" posle %1f \n",Omegaij[ i][ j] );
      }
      }
      CoolCycle = true;
```

```
      for (int nf = 0; nf < FileNumbers; nf++)
      {
      for (int np = 0; np < ConnNumbers; np++)
      {
      // подача параметрів трафіку на вхід нейронної мережі
      for (int i = 0; i < input; i++)
      nnin[ i] = FData[ nf][ np][ i] ;
      _calculationLVQ();
      CoolCycle = (CoolCycle && (nnout[ 0] == Etalon[ nf][ np][ 0] ));

      }
      }
      }
      else
      {
      CoolCycle = true;
      for (int i = 0; i < hid; i++)
      {
      for (int j = 0; j < input; j++)
      Omegaij[ minindex][ j] = Omegaij[ minindex][ j]  - 1/t *  (nnin[ j]  -
Omegaij[ minindex][ j] );

      }
      }
      InCycle = InCycle - 1;
      }
      }
      printf ("\n Iteration %d", t);
      t = t+1;
getch();
}
if (t < 49) all = all+1;
printf ("\n Count %d \n", all);
}
}

int _tmain(int argc, _TCHAR* argv[])
{
      int i = 10;
      double fp = 1.5;
      char s[] = "this is a string";
      char c = '\n';

      struct _finddata_t c_file;
      intptr_t hFile;

      srand( (unsigned)time( NULL ));

      fopen_s( &stream, "d:/datafile.dat", "wt" );
      fprintf_s( stream, "%s%c \n", "ConnectionNormal      Attack");

      _learning (); // процедура навчання детектора

      stBufFile _BufFile; // читання даних для аналізу


      if( (hFile = _findfirst( "D:/AIS/*.*", &c_file )) == -1L )
      printf( "No *.* files in current directory!\n" );
      else
      {
      do
      {
```

```
if (strcmp(c_file.name,".")&&strcmp(c_file.name,".."))
{
printf( " %s \n", c_file.name);

int fff = 0;
while (PathCheck1[ fff] !='\0' )
{
PathTemp[ fff] = PathCheck1[ fff] ;
fff++;
}
PathTemp[ fff] = '\0' ;
int ff = 0;
while (c_file.name[ ff] !='\0' )
ff++;

for (int f = 0; f < ff; f++)
{
PathTemp[ fff + f] = c_file.name[ f] ;
PathTemp[ fff + f+1] = '\0' ;
}

_BufFile = ReturnBufFile(PathTemp);

double SelfValue = 0;
double VirusValue = 0;
for (int ii = 0; ii < _BufFile.iSize - input; ii++)
{
for (int j = 0; j < input; j++)
nnin[ j] = _BufFile.cBuf[ ii+j] ;
_calculationLVQ ();
SelfValue += nnout[ 0] ;
NonSelfValue += nnout[ 1] ;
}
SelfValue /= _BufFile.iSize -input;
NonSelfValue /= _BufFile.iSize -input;

printf("\t Normal %1f \t",SelfValue);
printf(" Attack %1f \n",NonSelfValue);

fprintf_s( stream, "%60s%c", c_file.name);
fprintf( stream, "%15f", SelfValue);
fprintf( stream, "%15f\n", NonSelfValue);
}
}
while( _findnext( hFile &c_file )== 0 );
_findclose( hFile );

}



if( (hFile = _findfirst( "D:/KDD/*.*", &c_file )) == -1L )
printf( "No *.* files in current directory!\n" );
else
{
do
{
if (strcmp(c_file.name,".")&&strcmp(c_file.name,".."))
{
printf( " %s \n", c_file.name);

int fff = 0;
while (PathCheck[ fff] !='\0' )
```

```
{
PathTemp[ fff] = PathCheck[ fff] ;
fff++;
}
PathTemp[ fff] = '\0' ;
int ff = 0;
while (c_file.name[ ff] !='\0' )
ff++;

for (int f = 0; f < ff; f++)
{
PathTemp[ fff + f] = c_file.name[ f] ;
PathTemp[ fff + f+1] = '\0' ;
}

_BufFile = ReturnBufFile(PathTemp);

double SelfValue = 0;
double NonSelfValue = 0;
for (int ii = 0; ii < _BufFile.iSize - input; ii++)
{
for (int j = 0; j < input; j++)
nnin[ j] = _BufFile.cBuf[ ii+j] ;
_calculationLVQ ();
SelfValue += nnout[ 0] ;
NonSelfValue += nnout[ 1] ;
}
SelfValue /= _BufFile.iSize -input;
NonSelfValue /= _BufFile.iSize -input;

printf("\t Normal %1f \t",SelfValue);
printf(" Attack %1f \n",NonSelfValue);

fprintf_s( stream, "%60s%c", c_file.name);
fprintf( stream, "%15f", SelfValue);
fprintf( stream, "%15f\n", NonSelfValue);
}
}
while( _findnext( hFile &c_file )== 0 );
_findclose( hFile );

}

fclose( stream );
getch();
return 0;
}
```

Додаток М

Акти впровадження результатів дисертаційного дослідження

«Утверждаю»

Директор ООО «СофтИнвест»

_____ Шевеленков В. В.

«___» _____ 2012 р.

АКТ

о внедрении результатов диссертационной работы

Комара Мирослава Петровича

Комиссия в составе:

Технический директор Ерошенко Я. Ю.

Начальник отдела разработки
программного обеспечения Емельянов А. Ю.

Начальник отдела телекоммуникаций Ануфриев Ю. А.

Составила настоящий акт о внедрении результатов диссертационной работы соискателя ученой степени кандидата технических наук Тернопольского национального экономического университета Комара М.П. на ООО «СофтИнвест» (г. Брест, Республика Беларусь) в том, что он проводил работы по внедрению системы обнаружения и классификации атак на информационные телекоммуникационные сети.

В процессе решения соискателем научно-практической задачи повышения достоверности обнаружения компьютерных атак на основе теории искусственных нейронных сетей и искусственных иммунных систем,

Комаром М.П. были получены лично и использованы на ООО «СофтИнвест» такие результаты научных исследований и практические результаты:

- нейросетевые методы обнаружения и классификации атак на информационные телекоммуникационные сети, суть которых заключается в построении нейросетевого детектора классификации информации в каналах обмена данными компьютерных сетей и построении совокупного классификатора для иерархический классификации компьютерных атак на основе нейросетевых детекторов;

- комбинированный метод обнаружения и классификации атак на информационные телекоммуникационные сети, суть которого заключается в интеграции нейросетевых детекторов в искусственную иммунную систему, что позволило сформировать лучшую популяцию детекторов для каждого типа компьютерной атаки;

- система обнаружения и классификации атак на информационные телекоммуникационные сети, которая позволила эффективно обнаруживать как известные, так и неизвестные компьютерные атаки, а также повысить достоверность обнаружения компьютерных атак.

Ерошенко Я. Ю.

Емельянов А. Ю.

Ануфриев Ю. А.

«ЗАТВЕРДЖУЮ»
Проректор з наукової роботи
Тернопільського національного
економічного університету
проф. Задорожний З.М. В.

10 2012 р.

АКТ
про впровадження результатів дисертаційної роботи аспіранта кафедри інформаційно-обчислювальних систем та управління (ІОСУ) Комара Мирослава Петровича
у науково-дослідній роботі на тему:
«Методи та засоби виявлення вторгнень на комп'ютерні системи»
(номер державної реєстрації роботи – 0110U000786)

Ми, комісія у складі директора Науково-дослідного інституту інтелектуальних комп'ютерних систем Тернопільського національного економічного університету (ТНЕУ) к.т.н., професора кафедри ІОСУ Кочана В.В., наукового керівника науково-дослідної роботи, д.т.н., професора Саченка А.О. та начальника науково-дослідної частини ТНЕУ Письменного В.І., створена для приймання роботи, виконаної в рамках тематичного плану науково-дослідних робіт ТНЕУ на тему «Методи та засоби виявлення вторгнень на комп'ютерні системи» (номер державної реєстрації – 0110U000786), встановила:

1. Розроблена Комаром М.П. система виявлення та класифікації атак базується на запропонованих у його дисертаційній роботі методі побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі, методі побудови сукупного класифікатора для ієрархічний класифікації комп'ютерних атак та комбінованому методі виявлення та класифікації атак.
2. Проведені за участю Комара М.П. експериментальні дослідження дали змогу здійснити перевірку запропонованої інтелектуальної інформаційної технології в рамках тестової комп'ютерної системи виявлення та класифікації вторгнень на комп'ютерні системи.
3. Експериментальні дослідження показали, що розроблена інтелектуальна інформаційна технологія дозволяє виявляти і класифікувати атаки на інформаційні телекомунікаційні мереж із похибкою, яка не перевищує 0,53% на оціночному наборі атак.

Директор НДІ інтелектуальних
комп'ютерних систем,
к.т.н., професор кафедри ІОСУ — Кочан В.В.

Науковий керівник
науково-дослідної роботи,
д.т.н., професор — Саченко А.О.

Начальник науково-дослідної
частини ТНЕУ — Письменний В.І.

«УТВЕРЖДАЮ»
Проректор по научной работе
Брестского Государственного
технического университета,
к.ф.-м.н., доцент Губанов В.С.
_____
«___» _____ 2012 г.

АКТ
о внедрение результатов диссертационной работы Комара Мирослава Петровича на тему: «Интеллектуальная информационная технология обнаружения и классификации атак на информационные телекоммуникационные сети» в рамках двухстороннего договора о партнерстве, сотрудничестве и научном обмене между Тернопольским национальным экономическим университетом (Украина) и Брестским Государственным техническим университетом (Республика Беларусь)

Этим актом нижеподписавшиеся Головко В.А., заведующий кафедрой интеллектуальных информационных технологий, Ракецкий В.М., декан факультета электронно-информационных систем, Сторожук Н.Ю., начальник учебно-методического отдела Брестского Государственного технического университета (БрГТУ), г. Брест, Республика Беларусь, подтверждают, что Комар М.П. принимал активное участие в научных работах в рамках двухстороннего договора о партнерстве, сотрудничестве и научном обмене между Тернопольским национальным экономическим университетом (Украина) и Брестским Государственным техническим университетом (Республика Беларусь).

Комар М.П. провел экспериментальные исследования по теме диссертационной работы на кафедре интеллектуальных информационных технологий в лаборатории искусственных нейронных сетей под руководством д.т.н., профессора Головко В.А. За результатами провиденных экспериментов опубликовано пять научных работ в сборниках международных конференций.

Разработанные в диссертационной работе Комара М.П. методы, были использованы при создании нейросетевой системы обнаружения и классификации вторжений в лаборатории искусственных нейронных сетей кафедры интеллектуальных информационных технологий БрГТУ. Разработанное алгоритмическое и программное обеспечение было протестировано на тестовых наборах данных на базе разработанного макета в режиме реального времени.

Результаты диссертационной работы Комара М.П. внедрены и используются в учебном процессе факультета электронно-информационных систем БрГТУ для студентов специальностей «Автоматизированные системы обработки информации» и «Искусственный интеллект» при преподавании дисциплин: «Нейросетевые методы обработки информации», «Интеллектуальные технологии обработки данных», «Современные методы защиты компьютерных сетей».

Начальник учебно-методического отдела Н.Ю.Сторожук

Декан факультета электронно-информационных систем, к.ф.-м.н., доцент В.М.Ракецкий

Заведующий кафедрой интеллектуальных информационных технологий, д.т.н, профессор В.А.Головко

«ЗАТВЕРДЖУЮ»
Проректор з науково-педагогічної роботи
Тернопільського національного
економічного університету

_____ Шинкарик М.І.
_____ 2012 р.

АКТ

про впровадження в навчальний процес Тернопільського національного економічного університету (ТНЕУ) результатів дисертаційної роботи аспіранта кафедри інформаційно-обчислювальних систем та управління Комара Мирослава Петровича «Інтелектуальна інформаційна технологія виявлення та класифікації атак на інформаційні телекомунікаційні мережі»

Ми, комісія у складі: декана факультету комп'ютерних інформаційних технологій, д.т.н., проф. Дивака М.П., завідувача кафедри інформаційно-обчислювальних систем та управління, д.т.н., проф. Сачeнка А.О. та завідувача кафедри комп'ютерної інженерії, д.т.н., доц. Березького О.М. склали цей акт про те, що результати дисертаційної роботи Комара М.П. впроваджені та використовуються в навчальному процесі факультету комп'ютерних інформаційних технологій ТНЕУ для студентів напрямів підготовки 6.050102 «Комп'ютерна інженерія» та 6.050101 «Комп'ютерні науки». Зокрема при викладанні дисциплін: «Комп'ютерні мережі», «Штучні нейронні мережі», «Телекомунікаційні системи», «Методи та системи штучного інтелекту» використано:

- метод побудови нейромережевого детектора атак на інформаційні телекомунікаційні мережі;
- метод побудови сукупного класифікатора для ієрархічний класифікації комп'ютерних атак;
- комбінований метод виявлення і класифікації атак на інформаційні телекомунікаційні мережі;
- структура нейромережевої імунної системи виявлення і класифікації атак.

Ефект від використання результатів дисертаційної роботи Комара М.П. полягає у підвищенні якості вивчення майбутніми фахівцями сучасних методів виявлення та класифікації атак на інформаційні телекомунікаційні мережі, обробки інформації, навчання нейронних мереж, підвищення якості підготовки курсових та дипломних проектів, що в результаті забезпечує підвищення рівня підготовки фахівців з напрямів 6.050102 «Комп'ютерна інженерія» та 6.050101 «Комп'ютерні науки».

Декан факультету комп'ютерних
інформаційних технологій, д.т.н., професор _____ Дивак М.П.

Завідувач кафедри інформаційно-обчислювальних
систем та управління, д.т.н., професор _____ Сачeнко А.О.

Завідувач кафедри комп'ютерної інженерії,
д.т.н., доцент _____ Березький О.М.