МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Система виявлення кібератак. Інформаційна технологія функціонування системи виявлення атак в режимі моніторингу»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Довбиш А.С.**

Студентка групи КБ – 71 — **Теницька А.О.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студентки четвертого курсу, групи КБ-71 спеціальності "Кібербезпека" денної форми навчання Теницької Альони Олексіївни.

Тема: "Система виявлення кібератак. Інформаційна технологія функціонування системи виявлення атак в режимі моніторингу "

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) аналіз проблеми дослідження ; 2) опис методу дослідження; 3) інформаційне, алгоритмічне та програмне забезпечення системи виявлення атак.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Довбиш А.С.

Завдання прийняв до виконання _____ Теницька А.О.

РЕФЕРАТ

Записка: 67 стор., 6 рис., 7 табл., 1 додаток, 24 джерела.

Мета роботи — реалізація лінійного алгоритму інформаційно-екстремального машинного навчання системи виявлення атак та перевірка функціональності в режиму моніторингу.

Об'єкт дослідження — процес виявлення кібератак.

Предмет дослідження — метод інформаційно-екстремального машинного навчання системи виявлення атак.

Результати — розроблено алгоритм та програмне забезпечення системи виявлення атак в рамках інформаційно-екстремальної інтелектуальної технології аналізу даних. Розглянуто та реалізовано базовий алгоритм інформаційно-екстремального машинного навчання. Проведено перевірку функціонування системи виявлення атак в режимі моніторингу. Алгоритм розроблено за допомогою MATLAB – пакету прикладних програм для вирішення завдань технічних обчислень.

СИСТЕМА ВИЯВЛЕННЯ КІБЕРАТАК, ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ, МАШИННЕ НАВЧАННЯ, КАТЕГОРІЙНА МОДЕЛЬ, БАЗОВИЙ АЛГОРИТМ МАШИННОГО НАВЧАННЯ, РЕЖИМ МОНІТОРИНГУ, ВХІДНИЙ МАТЕМАТИЧНИЙ ОПИС

ЗМІСТ

ВСТУП

Значний розвиток комп'ютерних мереж та цифризація всіх галузей військової, економічної, соціальної, оборонної та інших сфер зумовили зростання кількості атак на інформаційні системи.

Під кібератакою розуміють навмисні дії в кіберпросторі, які здійснюються задля досягнення таких цілей, як отримання несанкціонованого доступу до систем та порушення конфіденційності, доступності і цілісності інформації, яка обробляється, передається та зберігається в технологічних системах [1]. Слід відзначити, що з кожним роком атаки стають все масштабнішими та досконалішими.

В останні роки інформація розглядається як критично важливий державний ресурс і тому потребує все більшої уваги до її захисту. Згідно звіту національної поліції України про результати роботи у 2020 році було зареєстровано понад 5 тисяч кіберзлочинів з матеріальними збитками на суму 241 млн грн. [2].

Розглядаючи статистику останніх років можна зробити висновок, що, незважаючи на існування великої кількості механізмів для захисту інформації, кількість злочинів, пов'язаних з кібербезпекою, різко зростає. Тому, виявлення різного роду мережевих атак або несанкціонованих дій та захист від них є одним з найактуальніших та найважливіших питань в даний час.

Останніми роками все більше і більше уваги приділяють досить перспективним напрямкам у даній галузі, до яких належить технологія блокчейн, квантова криптографія та система виявлення атак (СВА). Особливі надії покладаються на останню, так як це новий та перспективний напрямок у сфері захисту інформації. Основним призначенням СВА є виявлення вторгнень або ж спроб несанкціонованого доступу. Серед методів системи виявлення атак вирізняють сигнатурний та метод виявлення аномалій. Наразі, підвищена увага приділяється реалізації системи, яка б об'єднувала зазначені напрямки. На

сьогодні дана ідея реалізується за допомогою використання штучних нейронних мереж (ШНМ), але вони мають вже відомі всім суттєві недоліки.

Новим та перспективним способом у даній галузі є використання методів та ідей інформаційно-екстремальної інтелектуальної технології аналізу даних (IEI-технології), основною метою якої є максимізація інформаційної спроможності системи виявлення атак у процесі машинного навчання.

Метою даної роботи є аналіз сучасного стану та тенденцій розвитку системи виявлення атак, огляд існуючих програмних рішень, які позиціонують себе як СВА, а також реалізація лінійного алгоритму інформаційно-екстремального машинного навчання системи виявлення атак та перевірка системи в режимі екзамену.

1 АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

1.1 Сучасний стан та тенденції розвитку системи виявлення атак

Система виявлення атак (СВА, Intrusion Detection System, IDS) — програмний або апаратний засіб, який розроблений для виявлення несанкціонованого доступу в мережу або систему, а також для визначення спроб несанкціонованого управління ними через Інтернет. [3]

СВА гарантують виявлення [4]:

- дій шкідливого ПЗ (комп’ютерних вірусів, тощо).
- неавторизованого доступу до системи та інформації;
- мережевих атак на вразливі сервіси;
- атак, націлених на підвищення прав користувачів.

Використання СВА допомагає у досягненні таких цілей та вирішенні таких проблем, як:

- виявлення вже відомих мережевих атак та вторгнень;
- прогнозування можливих майбутніх атак та виявлення вразливостей для уникнення їх подальшого розвитку;
- визначення правильного розташування джерела атаки по відношенню до мережі (внутрішні або зовнішні атаки);
- аналіз та моніторинг мережевої, користувальницької та системної активності;
- отримання корисної інформації про проникнення, для відновлення після вторгнення та налаштування конфігурації мережі;
- забезпечення належного контролю якості адміністрування, особливо у складних та великих мережах;
- встановлення автоматичного контролю за всіма компонентами інформаційної системи, що автоматично знижує навантаження на персонал та унеможливлює людських фактор похибки.

На рис. 1.1 відображена типова архітектура систем виявлення атак [4].

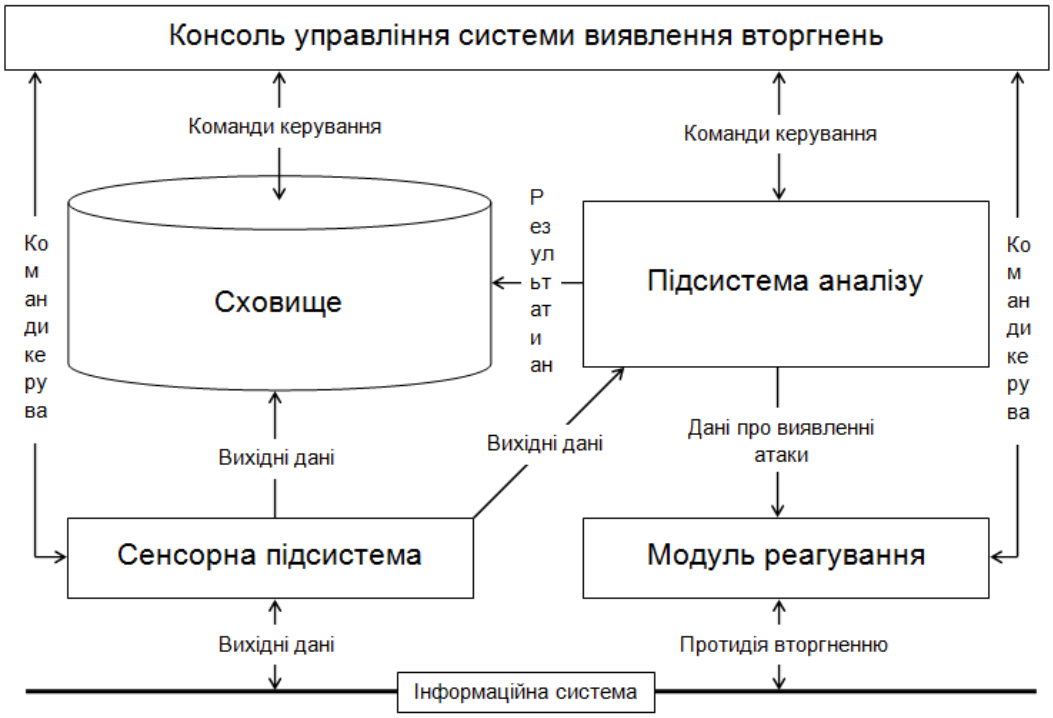


Рисунок 1.1 – Схема типових систем виявлення атак

Як видно з рис. 1.1, до основних компонентів СВА належать консоль управління, сховище, підсистема аналізу, сенсорна підсистема та модуль реагування.

Консоль управління забезпечує конфігурування системи виявлення атак, перегляд інцидентів несанкціонованих вторгнень, які виявлені підсистемою аналізу і спостереження за станом інформаційної системи, мережі та СВА.

У сховищі даних збираються результати аналізу та база первинних подій.

Сенсорна підсистема виявлення атак призначена для накопичення подій, які пов’язані з безпекою системи або мережі, що підлягає захисту.

Підсистеми аналізу є основним модулем системи виявлення атак. Її призначенням є вивчення інформації, яка надходить від сенсорної підсистеми. За результатами аналізу ідентифікує атаки або ж підозрілі дії.

Модуль реагування здійснює протидію виявленим вторгненням у систему або ж несанкціонованому доступу за визначеними інструкціями.

Класифікація систем виявлення атак є досить широкою та включає в себе такі позиції, як [5]:

- класифікація за методами виявлення атак;
- класифікація за способом моніторингу;
- класифікація за архітектурою: встановлені за межами системи та встановлені на самій системі;
- класифікація за характером відповіді: активні та пасивні;
- класифікація за принципом роботи: статичні і динамічні;
- класифікація за часом реакції: реального часу та пакетного часу;
- класифікація за джерелом аудиту: мережеві пакети, дані від сенсорів СВА, аналіз стану системи, дані журналів додатку та шляхів аудиту;
- класифікація за джерелом збору даних: централізовано або у розподіленому режимі;
- класифікація за парадигмою виявлення: ті, які оцінюють переходи між станами та ті, які оцінюють стан;
- класифікація за технологіями побудови: бездротові технології та дротові.

Найпопулярнішими та найважливішими при виборі СВА традиційно вважають тільки дві характеристики, до яких відносять методи виявлення атак та середовище моніторингу. Саме на них буде звернена увага у випускній роботі.

За способами моніторингу СВА розподіляють на 2 типи. Так як будь-яка атака може бути виявлена або на базі хосту, або під час аналізу мережевого трафіку, то виділяють network-based (Network Intrusion Detection Systems, NIDS) і host-based (Host Intrusion Detection Systems, HIDS).

1.1.1 Мережева система виявлення атак

Мережева система виявлення атак (Network intrusion detection system, NIDS) - система виявлення атак, яка призначена для допомоги організаціям у

моніторингу своїх хмарних, локальних і гібридних середовищ на предмет підозрілих подій, які можуть вказувати на компрометацію. Сюди входять порушення політики і сканування портів, а також невідомий вихідний і цільовий трафік [6].

NIDS переглядає усі пакети, які потрапляють до системи на існування в них різних підозрілих ознак. Якщо, наприклад, виявлено велику кількість запитів на TCP з'єднання з широким діапазоном різних портів, то, найімовірніше, проводиться сканування портів.

NIDS не обмежується відстеженням тільки вхідного мережевого трафіку. Часто важливу інформацію про те, що відбувається атака можна отримати також з вихідного або локального трафіку. Дія деяких атак може розвиватися усередині мережі, яка спостерігається або сегмента мережі, і ніяк не відображатися на вхідному трафіку.

Найчастіше, NIDS добре взаємодіє з іншими захисними системами. Мережева система виявлення вторгнень може на основі результатів своєї роботи оновлювати чорні списки міжмережевих екранів, заносячи туди IP адреси машин, запідозрених у здійсненні атаки.

До переваг такої системи відносять [4]:

- централізоване управляння як наслідок великого покриття моніторингу;
- NIDS не позначаються на продуктивності існуючої інформаційної системи;
- вказані СВА, здебільшого, вважаються пасивними пристроями, які можуть перехоплювати мережевий трафік, не навантажуючи при цьому мережу різними службовими потоками.

Незважаючи на суттєві переваги NIDS, можна виділити ряд не менш важливих недоліків:

- в мить високого навантаження на систему дані СВА не здатні розпізнати напад;

- NIDS не спроможні аналізувати зашифровану інформацію;
- під час використання NIDS не аналізується ступінь проникнення, лише повідомляється про вчинений напад;
- вимагають додаткового та більш ретельного налаштування і функціональності різних мережевих пристроїв;
- NIDS не спроможні розпізнати результат атаки. Вони мають змогу тільки визначити, що атаку було розпочато, але не вказують на її успішність. Це свідчить про те, що після того, як NIDS повідомить про можливий напад, адміністратору необхідно самостійно досліджувати кожен хост на можливість реального проникнення.

1.1.2 Хостова система виявлення атак

Хостова система виявлення атак (Host-based intrusion detection system, HIDS) – це система виявлення атак, яка спостерігає та аналізує події, що відбуваються тільки всередині системи (на відміну від NIDS, яка відстежує в першу чергу мережевий трафік) [7].

Метою хостовой СВА є стеження за всіма подіями, що відбуваються в комп'ютерній системі і перевірка їх на відповідність моделі безпеки. Поки мережева СВА контролює мережеві пакети, що проходять через систему, хостова СВА, у свою чергу, перевіряє, яка програма має доступ до яких ресурсів, і може виявити, що, наприклад, текстовий процесор раптово почав змінювати базу даних системних паролів. HIDS просто відстежує поточний стан системи, зберігає інформацію (як в оперативній пам'яті, так і у файловій системі), за даними системних логів і перевіряє, наскільки цей стан відповідає «нормальному».

Можна сказати, що HIDS є агентом, який гарантує, що ніхто (за межами системи або всередині системи) не може порушити політику безпеки, встановлену операційною системою.

Завдяки своєму вигідному розташуванню безпосередньо на сервері, HIDS пропонує значні переваги [4]:

- не вимагають додаткових функціональних можливостей мережевих пристроїв;
- можуть виявляти атаки, які NIDS не бачить;
- мають можливість моніторити події локально щодо хоста;
- працюють в мережі із зашифрованими даними, коли інформація знаходиться у загальнодоступному перегляді на сервері перед відправкою її клієнту.

Однак через особливості операційних систем, способів вторгнень та розташування систем виявлення атак можна виділити наступні недоліки:

- механізми збору інформації повинні бути встановленими і підтримуватися на кожному сервері, який слід контролювати;
- HIDS мають змогу бачити тільки мережеві пакети, які отримують від сервера, на якому встановлені, тому вони не можуть контролювати ситуацію по всій мережі;
- можуть бути атакованими та заблокованими навченим супротивником;
- знижують ефективність роботи контрольованого сервера, так як використовують його обчислювальні ресурси.

Розробки у галузі створення системи виявлення атак розпочалися у 1990-х роках. На сьогоднішній день відома безліч як комерційних, так і безкоштовних рішень у даній сфері. У цій роботі розглянуто найпоширеніші та найвідоміші останнім часом СВА.

1.1.3 Snort

Snort є однією з найпопулярніших безкоштовних мережевих систем виявлення атак з відкритим кодом. Розроблена в США компанією Sourcefire та Cisco. Працює на ОС Windows, Linux та Unix.

Ця IDS виявляє наступне [8]:

- Поганий трафік
- Використання експлойтів (виявлення Shellcode)
- Сканування системи (порти, ОС, користувачі і т.д.)
- Атаки на такі служби як Telnet, FTP, DNS, і т.д.
- Атаки DoS / DDoS
- Атаки пов'язані з Web серверами (cgi, php, frontpage, iss і т.д.)
- Атаки на бази даних SQL, Oracle і т.д.
- Атаки за протоколами SNMP, NetBios, ICMP
- Атаки на SMTP, imap, pop2, pop3
- Різноманітні Backdoors
- Web-фільтри
- Віруси

Всі виявлені загрози (список параметрів подачі тривоги має тонкі настройки), записуються в лог-файл. Snort працює за принципом аналізу пакетів транспортного рівня, тому для його використання, потрібен переклад мережевої карти в спеціальний моніторний режим. Розробники враховували проблему споживання системних ресурсів системами класу IDS, тому Snort невимоглива до заліза і працює у фоновому режимі [9].

Snort підтримує 3 режими роботи:

- режим сніфферу;
- режим реєстратора пакетів;
- режим системи виявлення атак.

У режимі сніферу можна отримати просто зчитування пакетів, які проходять по мережі. Тобто просто система перехвачує і виводить на екран перехвачені дані.

У режимі реєстрації пакетів попередньо описаний режим – удосконалюється. Тобто інформація про перехвачені дані переглядається та реєструється у відповідний файл.

У разі використання останнього, третього, режиму системи виявлення атак обирається модуль аналізу, який застосовує набір правил для трафіку. Ці правила називаються «базовими політиками», їх можна завантажити з веб-сайту Snort. Однак, система надає можливість написати свої власні правила [10].

Архітектура системи абсолютно проста та складається с таких компонентів:

- декодера пакетів;
- ядра виявлення;
- підсистеми оповіщення та реагування.

Декодер здійснює набір процедур для послідовної декомпозиції пакетів згідно з рівнем мережевого стека. Таким чином, прийнятий кадр послідовно трансформується в пакет, сегмент і блок даних, враховуючи специфічні атрибути сигнатур для даного рівня. Підтримуються протоколи канального рівня Ethernet, SLIP, PPP, а також ATM.

Ядро об’єднує існуючі правила в ланцюги, що складають відповідні двомірні послідовності, за якими здійснюється проходження кожного пакету. Підсистема оповіщення та реагування відповідає за збереження результатів аналізу трафіку у відповідних журналах Snort або передачу цієї інформації системними службами реєстрації подій ОС [11].

1.1.4 OSSEC

OSSEC (Open Source Host-based Intrusion Detection System) – це широкомасштабна багатоплатформена хостова система виявлення атак (HIDS) з відкритим вихідним кодом.

OSSEC інтегрує моніторинг цілісності файлів, аналіз журналів, моніторинг реєстру Windows, централізоване застосування політики, виявлення руткітів та сповіщення в режимі реального часу. Вона працює в більшості операційних систем, включаючи Linux, OpenBSD, FreeBSD, MacOS, Solaris та Windows [12].

OSSEC формується з трьох елементів, до яких відносять:

- базову програму – пропонує базові функціональні можливості, тобто сервер для збирання та аналізування даних, який установлюється в Unix системах;
- агента Windows – маленьких програм, які встановлені для моніторингу системи;
- веб-інтерфейсу – дозволяє слідкувати за попередженнями у зручному для адміністратора вигляді.

Агенти використовуються для збору інформації з віддалених систем, хоча можна працювати і без нього (наприклад, на деяких маршрутизаторах). Порт 1514 UDP використовується для зв'язку між агентом і сервером, з'єднання здійснюється через зашифрований канал SSH. Отримані дані опрацьовуються на сервері на основі правил в форматі XML, і, у разі виявлення спроб модифікації реєстру та важливих файлів (аналізуються права доступу, SHA1 та MD5), і інших потенційно небезпечних дій – система відображає відповідне повідомлення, а при високому рівні намагається відреагувати на спробу злому.

Деякі правила можуть брати умови з інших правил і базувати свої дії на них. Наприклад, кілька записів з id – 31105 з різних IP-адрес вказують на розподілену XSS атаку. Регулярні вирази можна використовувати в правилах, тому маючи певні навички можна описати майже будь-яку ситуацію.

Система дозволяє адміністратору легко вказати на цікаві події та отримати відповідні повідомлення через сервіси SMTP та SMS, а також налаштувати відповідь, блокуючи віддалений вузол.

Для того щоб зменшити ймовірність непередбачуваного блокування своїх ресурсів в разі помилки OSSEC, розробники пропонують готові структури, використовують тайм-аут, після якого ресурс розблоковується та білі списки. За замовчуванням сповіщення зберігаються у текстових файлах, однак для реалізації цієї мети також можливо підключити базу даних MySQL [13].

1.1.5 Suricata

Suricata – програмний засіб для реалізації IDS, IPS (система запобігання атак) та NSM (моніторинг мережевої безпеки). Продукт був розроблений організацією Open Information Security Foundation (OISF) як альтернатива Snort. Сумісна з багатьма операційними системами, такими як Windows, Linux (Debian, CentOS, Ubuntu, Red Hat), FreeBSD, MacOS X. Suricata може працювати як самостійна система, так і в якості додатка до деяких систем [14].

Програмний засіб працює на рівні додатків. Suricata розуміє такі протоколи, як HTTP, FTP та SMB. Він також відслідковує і протоколи нижчого рівня, такі як TCP, UDP, TLS та ICMP. Suricata дозволяє виявити загрози які приховані в звичайних запитах, а також реалізована можливість вилучання файлів, що дозволяє адміністраторам самостійно перевіряти підозрілі файли.

Архітектура Suricata дуже добре зроблена і допомагає розподілити робоче навантаження між кількома ядрами процесора та потоками для досягнення оптимальної роботи. Наприклад, для пристроїв, чия відеокарта в основному знаходиться в неактивному стані, частина навантаження може бути перекладена на неї [15].

Suricata містить такі модулі:

- захоплення;

- збір;
- декодування;
- виявлення;
- виведення.

Стандартно захоплений трафік прямує одним потоком до декодування. Звісно, це краще рішення для детектування, але створює велике навантаження на систему. Однак, у налаштуваннях програмного засобу можна відразу після захоплення розподілити потоки та визначити спосіб розподілу потоків між процесорами. Це пропонує численні можливості для підвищення ефективності обробки трафіку на даному обладнанні в даній мережі [16].

У разі порушення хоча б одного налаштованого правила – Suricata здійснює оперативну реакція на атаку, маркуючи отримані пакети данних [10]:

- NF_DROP (заборонений доступ);
- NF_ACCESS (наданий доступ);
- NF_REPEAT (пакети відповідним чином маркуються та повторно відправляються на правила брандмауера, який вирішує подальше призначення отриманого пакету).

1.1.6 Zeek

Zeek (до 2018 року носила назву Bro) – відкрита, безкоштовна система для виявлення мережевих атак. В ній надані модулі для аналізу різних мережевих протоколів, які функціонують на рівні додатків. Система дозволяє записувати всю мережеву діяльність у спеціальний детальний журнал. Zeek оптимізована для використання в мережах з великою пропускною здатністю. Працює система на транспортному, мережевому та рівні додатків.

Функція виявлення вторгнень Zeek виконується в два етапи: реєстрація трафіку і аналіз.

Першим з інструментів аналізу є механізм обробки подій Zeek. Він відстежує активізацію подій, таких як нове TCP-з'єднання або HTTP-запит. Кожна подія реєструється, тому ця частина системи не залежить від політики – вона просто надає список подій, в яких аналіз може виявити повторення дій або різноманітну підозрілу активність, генеровану одним і тим самим обліковим записом користувача.

Аналіз даних, зібраних механізмом обробки подій виконується сценаріями політики. Стан попередження провокує дію, саме тому Zeek є системою запобігання вторгнень. Сценарії політики можна налаштувати, але вони, як правило, працюють за стандартною структурою, що включає зіставлення сигнатур та аналіз з'єднань. Кожна політика – це набір відповідних правил. Система не обмежує в кількості активних політик та в протоколах, що накладають додаткові шари, які можна перевіряти. Після виявлення атак система реагує по різному: оповіщає адміністратора, вносить повідомлення в журнал, виконує команди операційної системи [10].

Zeek дозволяє відстежувати активність HTTP, DNS і FTP, а також відстежувати трафік SNMP. На більш низьких рівнях можна стежити за DDoS атаками або виявити сканування портів.

Zeek можна встановити на Unix, Linux та Mac OS.

1.1.7 Sagan

Sagan – це безкоштовна хостова система виявлення вторгнень, яка є найкращою альтернативою OSSEC. Головною перевагою даної IDS є те, що вона сумісна з даними, зібраними такими NIDS, як Snort. Sagan можна вважати більшою мірою системою аналізу журналів, ніж IDS, попри наявність IDS-подібних функцій. Елементом, якого йому не вистачає, щоб зробити його самостійним NIDS, є модуль пошуку пакетів. Однак плюсом є те, що Sagan не потребує спеціального обладнання та має гнучкість для аналізу як журналів

хостів, так і даних мережевого трафіку. Цей інструмент повинен бути супутником інших систем збору даних, щоб створити повну систему виявлення вторгнень [10].

Sagan сумісний не тільки з Snort, він також поширюється і на всі інструменти, що можна інтегрувати з Snort. Серед них Anaval, Squil, BASE та Snorby. Даний HIDS можна встановити на такі операційні системи: Linux, Unix та Mac-OS. Більше того, його можна подати за допомогою журналів подій Windows [18].

Однією, з важливих особливостей Sagan є те, що дана система включає IP – локатор, який дозволяє виявити географічне розташування IP – адрес, які система відзначила як підозрілі. Це дозволить виділити дії тих IP – адрес, які працюють спільно для вчинення однієї атак та вчасно зреагувати на виниклу загрозу.

Також, Sagan може розподілити свою обробку на декількох пристроях, що полегшить навантаження на процесор сервера.

Sagan включає виконання сценарію, що означає, що він буде генерувати попередження та виконувати дії щодо виявлення сценаріїв атак. Він може взаємодіяти з таблицями брандмауера для реалізації заборон IP у разі підозрілої діяльності з певного джерела.

Sagan не так часто розглядається як окрема IDS, однак його HIDS з комбінацією NIDS робить його цікавою пропозицією як компонента гібридного аналізу IDS.

1.1.8 Security Onion

Security Onion класифікується як безкоштовний NIDS, але він також включає функції HIDS. Система створена для операційної системи Linux з акцентом на управління журналами, моніторингом безпеки підприємства та виявлення атак. Вона контролює журнали та файли конфігурацій на наявність

підозрілих дій та перевіряє контрольну суму цих файлів на наявність несподіваних змін [10].

Мережевий аналіз проводиться за допомогою сніфера пакетів, який може відображати дані, які передаються на екран, а також записувати їх у файл.

Механізм аналізу у Security Onion значно ускладнений, так як дана система підтримує значну кількість інтерфейсних систем. Рекомендується використовувати Kibana, яка пропонує ряд гарних графіків та діаграм для полегшення розпізнавання статусу системи.

Більшість інструментів IDS є проектами з відкритим кодом. Розробники Security Onion вдало скористалися цим та створили дану систему, поєднавши елементи з OSSEC, Zeek, Snort та Suricata. Не зважаючи на те, що Security Onion написаний для роботи на Ubuntu, він інтегрує елементи інструментів аналізу та інтерфейсних систем. Серед них NetworkMiner, Snorby, Xplico, Sguil, ELSA та Kibana.

Система надає інформацію про стан пристроїв, а також про трафік.

1.1.9 SolarWinds Security Event Manager

Система SolarWinds Security Event Manager (SEM) розглядається як хостова система виявлення атак, оскільки вона займається управлінням файлами в системі. Однак, вона також управляє даними, зібраними в Snort, що робить її частиною мережевої системи виявлення атак [18].

Як було розглянуто раніше, Snort – це широко використовуваний пакетний сніфер з особливим форматом даних, який інші розробники IDS інтегрують у свою продукцію. SolarWinds Security Event Manager не є виключенням. У SolarWinds дані про трафік, які проходять по мережі, перевіряються за допомогою виявлення вторгнення в мережу. Для захоплення пакету використовується Snort, а SolarWinds , у свою чергу, застосовують для аналізу.

На додаток, розглядаємий IDS може отримувати мережеві дані від Snort в режимі реального часу, які є діяльністю NIDS.

Продукт SolarWinds також може виступати в якості системи запобігання вторгнень, оскільки він може ініціювати дії при виявленні атаки. У системі налаштовано більш ніж 700 правил кореляції подій, що дозволяє йому виявляти підозрілі дії і автоматично виконувати дії по виправленню ситуації. Ці дії називаються активними відповідями.

До вказаних активних відповідей відносять:

- сповіщення про інциденти через SNMP, екранні повідомлення або електронну пошту;
- ізоляція USB-пристроїв;
- призупинення дії облікового запису або виключення користувача;
- блокування IP-адреси;
- знищення процесів;
- завершення роботи або перезапуск системи;
- завершення роботи служби;
- запуск служби.

Можливості обробки повідомлень Snort в диспетчері подій безпеки роблять SolarWinds комплексним засобом моніторингу мережевої безпеки. Шкідливу активність можна зупинити майже миттєво завдяки здатності інструменту об'єднувати дані Snort з іншими подіями в системі. Ризик порушення роботи сервісу через виявлення помилкових спрацьовувань значно знижується завдяки точно налаштованим правилами кореляції подій.

Система працює на Windows Server, але він може реєструвати повідомлення, що генеруються комп'ютерами Unix, Linux і Mac OS, а також комп’ютерами з Windows [10].

1.1.10 Open WIPS-NG

Система Open WIPS-NG спеціально призначена для бездротових мереж. Назва «WIPS» означає «бездротова система запобігання вторгнень», тому цей NIDS одночасно виявляє та блокує атаки.

Система складається з трьох компонентів, до яких відносять:

- датчика;
- сервера;
- компонента інтерфейсу.

Датчик являє собою аналізатор пакетів, який також може управляти бездротовою передачею в середині потоку. Таким чином, датчик діє як приймач для системи.

Інформація, зібрана датчиком, пересилається на сервер, де відбувається чарівництво. Пакет серверних програм містить механізм аналізу, який виявляє зразки вторгнень. На сервері також створюються політики втручання для блокування виявлених вторгнень. Дії, необхідні для захисту мережі, відправляються датчику у вигляді інструкцій.

Інтерфейсний модуль системи являє собою приладову панель, яка відображає події та попередження для системного адміністратора. Тут також можна налаштувати параметри і відрегулювати або перевизначити захисні дії.

На даний момент кожна установка може включати тільки один датчик. Це не велика проблема, тому що можна вирішити кілька завдань за допомогою всього одного датчика. Але планується, що установка WIPS-NG дозволить контролювати кілька датчиків [10].

1.1.11 Samhain

Samhain - це безкоштовна хостова система виявлення вторгнень з відкритим кодом, що забезпечує контроль цілісності і моніторинг / аналіз логів

системи, а також виявлення руткітів (rootkit detection), контроль портів, виявлення програм з встановленим SUID і прихованих процесів

Його можна запустити на одному комп'ютері або на декількох хостах, пропонуючи централізований збір даних про події, виявлені агентами, що працюють на кожній машині.

Завдання, що виконуються кожним агентом, включають перевірку цілісності файлів, моніторинг файлів журналу та моніторинг портів. Процеси шукають руткіт-віруси, шахрайські SUID (права доступу користувачів) і приховані процеси. Система застосовує шифрування до обміну даними між агентами і центральним контролером в реалізаціях з декількома хостами. З'єднання для доставки даних файлу журналу включають вимоги аутентифікації, які не дозволяють зловмисникам захопити або замінити процес моніторингу.

Дані, зібрані Samhain, дозволяють аналізувати дії в мережі і виділяти попереджають ознаки вторгнення. Однак він не блокує вторгнення або усуває несанкціоновані процеси. Вам потрібно буде зберегти резервні копії ваших файлів конфігурації і ідентифікаторів користувачів, щоб вирішити проблеми, які виявляються монітором Samhain.

Одна з проблем, пов'язаних з проникненням хакерів і вірусів, полягає в тому, що зловмисник намагається сховатися. Це включає в себе відключення процесів моніторингу. Samhain використовує потаємну технологію, щоб приховати свої процеси, тим самим не дозволяючи зловмисникам маніпулювати або знищувати IDS. Цей прихований метод називається «стеганографія».

Центральні файли журналів і резервні копії конфігурації підписуються ключем PGP для запобігання злому зловмисниками.

Samhain сумісним з Unix, Linux і Mac OS. Центральний монітор буде збирати дані з різних операційних систем [10].

1.2 Методи виявлення атак

Запропоновані системи виявлення атак працюють завдяки виконанню двох вимог: або ж розуміння очікуваної поведінки суб’єкта системи, або ж знання всіх атак на систему та можливі їх модифікації. Тому за методами аналізу інформації та виявленням загроз виділяють сигнатурний метод та метод виявлення аномалій (anomaly-based IDS).

Вказані два методи опираються на твердження, що кожна атака може бути описана шляхом формування певного набору правил або за допомогою опису якоїсь формальної моделі.

Сигнатурний метод виявлення атак базується на описі вже відомих атак та порушень. У разі, якщо поведінка суб’єкта, підключеного до системи, збігається з описом атаки, яка вже відома – даний суб’єкт вважається зловмисником. СВА, яка працює за даним методом може досить ефективно виявити усі існуючі атаки, але майже не пристосована до нових, досі невідомих, атак. Тобто, якщо опис атаки не міститься у базі сигнатур – вона буде пропущена. Ще одним з важливих недоліків даного методу є те, що необхідно на постійній основі поповнювати базу знань сигнатур у разі виявлення нового виду атаки. У інакшому випадку це призведе до помилки другого роду, так званого «пропуску атаки». Існують випадки, коли СВА, розроблені за сигнатурним методом, можуть використовувати тільки певні сигнатури, що не допускає розпізнання варіантів вже відомих, загальних атак. Безумовно, даний метод вирізняється свою швидкодією та точність виявлення атак, які вже були описані.

Метод виявлення аномалій базується на тому, що для суб’єкта вказана певна нормальна поведінка. У разі відхилення від заданих правил поведінка суб’єкта вважається аномальною та він сприймається як загроза для системи. Головною метою даного методу є виявлення будь-якої підозрілої активності, яка відмінна від нормальної, а не чітке виявлення атак.

До підкатегорії даного методу можна віднести аналіз на основі профілів, тобто коли звичайна нормальна поведінка визначена для різних користувачів системи.

Можливими прикладами аномальної поведінки можуть бути: високі навантаження на процесор, досить велика кількість з'єднань з системою за відносно короткий проміжок часу або ж застосування сукупності зовнішніх пристроїв, які, як правило, не задіюються. Тобто будь яке відхилення від зазначеної поведінки суб'єкта буде вважатися атакою у випадку застосування аналізу на основі профілів. Однак відхилення від нормальної поведінки не завжди необхідно розцінювати як атаку.

Тому, основними недоліками методу виявлення аномалій можна вважати пропуск атаки та виникнення помилки першого роду, тобто хибне виявлення дії, яка не є атакою. Звісно, другий випадок буде значно небезпечніший.

Для вирішення проблеми виявлення аномалій слід вирішити дві основні задачі:

- визначення граничних значень атрибутів поведінки суб'єкта;
- побудова самого профілю суб'єкта, але це досить трудомістке завдання, яке потребує великої підготовчої роботи.

Безумовно, виявлення аномалій потребує постійної реєстрації всіх подій користувача системи, що значною мірою навантажує центральний процесор та споживає великі обсяги пам'яті для зберігання зібраних даних. Тобто, до ще одного недоліку можна віднести невелику швидкодію, що звичайно не підходить для систем, критичних до даного параметру.

Основною перевагою можна вважати те, що метод виявлення аномалій дозволить розпізнавати навіть досі невідомі атаки, при цьому не вимагаючи додаткової роботи над впровадженням нових сигнатур [19].

Порівняння двох зазначених методів вказане в таблиці 1.1:

Таблиця 1.1 – Порівняння методів виявлення атак

Характеристики	Метод виявлення аномалій	Сигнатурний метод
Ймовірність виявлення атаки	Обмежується налаштуваннями	Обмежується базою відомих атак
Ймовірність виникнення помилкового спрацювання	висока	низька
Ймовірність пропуску атаки	низька	середня
Обчислювальні витрати	великі	великі

1.3 Формалізована постановка інформаційного синтезу системи виявлення атак

З розвитком інформаційних та мережевих технологій зростає і складність інформаційних систем. Тому, постійно зростають вимоги до СВА, їх гнучкості, швидкодії та можливості виявлення більшої кількості атак. У зв’язку з зростанням ширини каналів зв’язку, зростає і кількість трафіку, що у свою чергу збільшує кількість сигналів тривоги. Тому, з’являється критична необхідність мінімізації хибних спрацювань СВА.

У зв’язку з цим останніми роками збільшується інтерес до побудови комбінованих систем виявлення атак, які зможуть поєднати у собі переваги розглянутих вищу методів виявлення атак.

Новітнім та перспективним способом розвитку системи виявлення атак є застосування методів та ідей інформаційно-екстремальної інтелектуальної технології аналізу даних (ІЕІ-технології), основною метою якої є максимізація інформаційної спроможності системи виявлення атак у процесі машинного навчання.

Виходячи з вище сказаного можна сформулювати формалізовану постановку інформаційного синтезу системи виявлення атак в рамках IEI-технології.

Задано алфавіт $\{X_m^o | m = \overline{1,M}\}$ класів розпізнання, який характеризує можливі трафіки інформаційної системи, та навчальну матрицю $\left\| y_{m,i}^{(j)} \right\|, i = \overline{1,N}, j = \overline{1,n}$, яка має тип «об'єкт – властивість», де N, n – кількість ознак розпізнання та реалізацій відповідних класів розпізнання. Зазначимо, що стовпчик матриці $\left\{ y_{m,i}^{(i)} | j = \overline{1,n} \right\}$ визначає навчальну вибірку значень i-ї ознаки, а рядок матриці $\left\{ y_{m,i}^{(i)} | j = \overline{1,N} \right\}$ – j-ту реалізацію. Концепція інформаційно-екстремальної інтелектуальної технології аналізу даних полягає в перетворенні вхідної навчальної матриці Y на робочу бінарну матрицю X, що у процесі машинного навчання способом допустимих перетворень адаптується до максимально повної імовірності прийняття правильних класифікаційних рішень. Тому, для бінарного простору Хемінга задається множина $\{g_m\}$ структурованих векторів параметрів функціонування, які впливають на функціональну ефективність машинного навчання системи виявлення атак. Вказані параметри називаються параметрами машинного навчання. Наведемо як структуру вектор параметрів машинного навчання СВА [20]:

$$g = \langle g_1, \ldots, g_{\xi 1}, \ldots, g_{\Xi_1}, f_1, \ldots, f_{\xi 2}, \ldots, f_{\Xi_2} \rangle, \Xi_1 + \Xi_2 = \Xi, , \qquad (1.1)$$

де $\langle g_1, \ldots, g_{\xi 1}, \ldots, g_{\Xi_1} \rangle$ - генотипні параметри функціонування, які впливають на параметри розподілу реалізацій класу розпізнання;

$\langle f_1, \ldots, f_{\xi 2}, \ldots, f_{\Xi_2} \rangle$ - фенотипні параметри функціонування ІС, які прямо впливають на геометрію контейнера класу розпізнавання.

При цьому задані обмеження на відповідні параметри машинного навчання:

$$R_{\xi 1}\left(g_1,\ldots,g_{\xi 1},\ldots,g_{\Xi_1}\right)\le 0, R_{\xi 2}\left(f_1,\ldots,f_{\xi 2},\ldots,f_{\Xi_2}\right)\le 0.$$

Потрібно [20]:

а) визначити оптимальні значення параметрів машинного навчання за формулою (1.1) $\left\{g_\xi^*|\xi=\overline{1,\Xi_1+\Xi_2}\right\}$, які забезпечують максимум інформаційного критерію, усередненого за алфавітом класів розпізнання:

$$\bar{E}^*=\frac{1}{M}\sum_{m=1}^{M}\max_{G_E\cap\{k\}}E_m^{(k)}, \tag{1.2}$$

де $\{k\}$ – впорядкована множина кроків машинного навчання;

$E_m^{(k)}$ - інформаційний критерій оптимізації параметрів машинного навчання, значення якого обчислюється на k- му кроці, для того щоб розпізнавати реалізації класу X_m^o;

G_E – робоча область, тобто допустима область визначення функції інформаційного критерію оптимізації.

б) побудувати шляхом допустимих перетворень у субпарацептуальному бінарному просторі ознак розпізнавання Хеммінга оптимальне чітке розбиття класів розпізнавання $\mathfrak{R}^{|M|}$ для апріорно класифікованого нечіткого розбиття $\widetilde{\mathfrak{R}}^{|M|}$, на основі якого будуть сформовані вирішальні правила;

в) при реалізації режиму екзамену для перевірки роботи ефективності машинного навчання необхідно прийняти рішення щодо належності реалізації образу, який розпізнається, до одного з класів заданого алфавіту $\{X_m^o\}$.

Отже, завданням роботи можна визначити оптимізацію параметрів функціонування за формулою (1.1) за відповідним інформаційним критерієм за формулою (1.2) в процесі інформаційно – екстремального машинного навчання. А також у режимі екзамену прийняти класифікаційне рішення за вирішальними правилами, побудованими на етапі навчання.

2 ОПИС МЕТОДУ ДОСЛІДЖЕННЯ

2.1 Основні положення інформаційно – екстремальної інтелектуальної технології аналізу даних

Основною ідеєю машинного навчання в контексті IEI-технології аналізу даних є перетворення апріорно узагальненого нечіткого розподілу простору ознак у чіткий розподіл класів розпізнавання шляхом оптимізації параметрів функціонування системи. У той же час відбувається цілеспрямований пошук глобального максимуму багато екстремальної функції статистично інформаційного критерію в робочій області її визначення, водночас відновлюючи оптимальні роздільні гіперповерхні, які будуються в радіальному базисі бінарного простору ознак розпізнавання [21].

Вирішальні правила в процесі оптимізації параметрів машинного навчання в контексті IEI-технології базуються на принципі відкладених рішень Івахненка О. Г. за мультициклічною ітераційною процедурою для визначення максимального граничного значення, яке усереднене за алфавітом класів розпізнавання інформаційного критерію оптимізації [22]:

$$g_{\xi}^{*} = \arg\max_{G_{\xi}} \left\{ \max_{G_{\xi-1}} \left\{ \dots \left\{ \max_{G_1 \cap G_E} \frac{1}{M} \sum_{m=1}^{M} E_m \right\} \dots \right\} \right\}, \quad (2.1)$$

де G_{ξ} – допустимий діапазон значень ξ -ї ознаки розпізнавання;

G_E - допустимий діапазон визначення функції інформаційного критерію для оптимізації параметрів машинного навчання;

E_m – інформаційний критерій оптимізації параметрів навчальної системи розпізнання реалізації класу X_m^0.

У той же час існують обмеження щодо алгоритму інформаційно-екстремального машинного навчання (2.1)

$$\left(\forall X_m^o \in \widetilde{\Re}^{|M|}\right)[X_m^o \neq \emptyset], \tag{2.2}$$

де $\widetilde{\Re}^{|M|}$ – розбиття простору ознак на відповідні класи розпізнавання, потужність якого становить $Card\widetilde{\Re} = M$;

$$\left(\exists X_k^0 \in \widetilde{\Re}^{|M|}\right)\left(\exists X_l^0 \in \widetilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \longrightarrow X_k^o \cap X_l^o \neq \emptyset]; \tag{2.3}$$

$$\left(\forall X_k^0 \in \widetilde{\Re}^{|M|}\right)\left(\forall X_l^0 \in \widetilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \longrightarrow KerX_k^o \cap KerX_l^o \neq \emptyset], \tag{2.4}$$

де $KerX_k^o$ – ядро класу розпізнавання X_k^o;

$KerX_l^o$– ядро класу розпізнавання X_l^o, який є найближчим до класу розпізнавання X_k^o;

$$\left(\forall X_k^0 \in \widetilde{\Re}^{|M|}\right)\left(\forall X_l^0 \in \widetilde{\Re}^{|M|}\right)\left[X_k^o \neq X_l^o \longrightarrow \begin{bmatrix}\left(d_k^* < d(x_k \oplus x_l)\right)\& \\ \&\,(d_l^* < d(x_k \oplus x_l)\end{bmatrix}\right], \tag{2.5}$$

де d_l^* - оптимальний радіус контейнера класу розпізнавання X_l^o;

$d(x_k \oplus x_l)$ – кодова відстань між вектором x_l класу розпізнавання X_l^0 та відповідним усередненим за ансамблем векторів ознак класу розпізнавання X_k^0 вектором x_k.

$$\bigcup_{X_m^o \in \Re} X_m^o \subseteq \Omega_B, \tag{2.6}$$

де Ω_B – є бінарним простором Хеммінга.

Відзначимо, що в формулах (2.3)–(2.6) $k \neq l$ та $k, l, m = \overline{1, M}$.

Глибина інформаційно-екстремального машинного навчання характеризується кількістю робочих параметрів систем, які оптимізовані за інформаційним критерієм. Внутрішній цикл процедури (2.1) реалізує базовий

алгоритм інформаційно-екстремального машинного навчання, який оптимізує геометричні параметри контейнерів класів розпізнавання.

Основною ідеєю машинного навчання за допомогою ІЕІ-технології, так як і у ШНМ є адаптація математичного вхідного опису систем розпізнавання до максимально повної ймовірності класифікаційних рішень. Принциповою відмінністю методів інформаційно-екстремального машинного навчання та нейроподібних структур є те, що вони розробляються як частина функціонального підходу до моделювання когнітивних процесів, властивих людям у процесі формування і прийняття класифікаційних рішень, тобто таким чином вони моделюють механізми природного інтелекту. А ось процеси машинного навчання трактують як оптимізацію параметрів системи розпізнавання, які мають вплив на її функціональну ефективність. Вказані параметри мають назву – «параметри машинного навчання». Будь-яка статистична інформаційна міра різноманітності аналізованих об’єктів може бути використана у якості критерію оптимізації в методах ІЕІ-технології [22].

У разі, якщо глибина машинного навчання в ШНМ визначається кількістю прихованих шарів, то в методах ІЕІ-технології вона визначається кількістю оптимізованих параметрів машинного навчання. У цьому випадку достатня глибина інформаційно-екстремального машинного навчання визначається за принципом відкладених рішень Івахненка О. Г. за умови, якщо досягнуто граничне максимальне значення інформаційного критерію оптимізації, усередненого за алфавітом класів розпізнавання. Вирішальні правила будуються за отриманими в процесі машинного навчання оптимальними геометричними параметрами контейнерів класів розпізнавання, які відновлюються в радіальній основі бінарного простору ознак Хеммінга. Побудова вирішальних правил в контексті геометричного підходу робить їх майже незмінними до багатовимірності простору ознак розпізнавання, оскільки сучасні комп’ютерні системи можуть обробляти двійкові вектори з 285 ознаками розпізнавання. Крім того, такі вирішальні правила відзначаються високою оперативністю прийняття

рішень у разі функціонування СВА в режимі екзамену, що є досить вагомим фактором для виявлення атак [22].

2.2 Категорійні моделі інформаційно – екстремального машинного навчання

При розробці алгоритмічного та інформаційного забезпечення системи, що навчається одним із основних етапів вважають категорійне моделювання інформаційно-екстремального машинного навчання.

Подається категорійна модель за базовим алгоритмом у вигляді орієнтованого графа відображення операторами одна на одну відповідних множин.

Вхідний математичний опис інформаційної системи, здатної навчатись представлено у вигляді структури [23]

$$\mathrm{I}_{\text{вх}} = \langle G, T, \Omega, Z, Y, X;\ f_1, f_2 \rangle, \tag{2.7}$$

де G – простір вхідних сигналів;

T – множина деяких моментів часу отримання інформації;

Ω – простір ознак розпізнавання;

Z – множина можливих станів кіберзахисту;

Y – вхідна навчальна матриця;

X – робоча бінарна навчальна матриця, яка адаптується в процесі інформаційно-екстремального машинного навчання до максимально можливої ймовірності прийняття точних діагностичних рішень;

f_1 – оператор для формування вхідної навчальної матриці;

f_2 – оператор для перетворення вхідної навчальної матриці Y в бінарну робочу навчальну матрицю X.

Категорійна модель інформаційно-екстремального машинного навчання СВА за базовим алгоритмом показана на рис. 2.1 [20] .

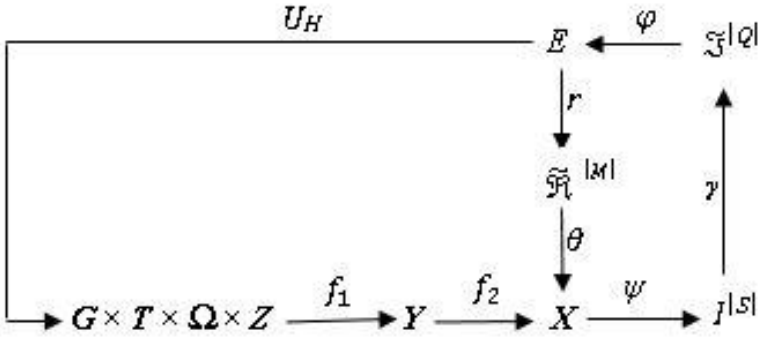


Рисунок 2.1 – Категорійна модель машинного навчання за базовим алгоритмом

Терммножина E, зображена на рис 2.1 , яка складається з значень інформаційного критерію, розрахованих на кожному етапі машинного навчання, є загальною для всіх контурів оптимізації параметрів. Оператор $r: E \to \widetilde{\Re}^{|M|}$ під час машинного навчання поновлює контейнери класів розпізнавання, які утворюють розбиття $\widetilde{\Re}^{|M|}$ в радіальному базисі бінарного простору ознак. Оператор θ відображає розбиття $\widetilde{\Re}^{|M|}$ на нечіткий розподіл апріорно класифікованих двійкових векторів ознак класів розпізнавання. Тоді оператор $\Psi: X \to I^{|S|}$, де $I^{|S|}$ є множиною гіпотез, перевіряє головну статистичну гіпотезу $\gamma_1: x_m^{(j)} \in X_m^o$. Оператор γ визначає набір точнісних характеристик $\Im^{|Q|}$, де $Q = S^2$, а оператор φ обраховує множину значень E інформаційного критерію оптимізації, що є функціоналом від точнісних характеристик. Контур оптимізації контрольних допусків замикається через терммножину D, елементами якої є значення контрольних допусків на ознаки розпізнавання. Оператор u регламентує процес машинного навчання [20].

У категорійну модель також можуть буть введені додаткові контури оптимізації параметрів, що впливають на ефективність машинного навчання задля максимізації інформаційної спроможності системи.

Отже, розглянута категорійна модель інформаційно-екстремального машинного навчання виступає узагальненою архітектурою базового алгоритму машинного навчання.

2.3 Опис базового алгоритму машинного навчання

У якості мети базового алгоритму навчання виділяють [22]:

- підвищення ефективності геометричних параметрів контейнерів класів розпізнавання;
- розрахунок інформаційного критерію для оптимізації параметрів машинного навчання;
- пошук глобального максимуму інформаційного критерію в відповідній робочій області визначення його функції.

Вхідними даними для навчання за базовим алгоритмом є трьохвимірний масив реалізацій класів розпізнавання $\left\{y_{m,i}^{(j)} | m = \overline{1,M}; i = \overline{1,N}; j = \overline{1,n}\right\}$; рівні селекції $\{\rho_m\}$ координат усереднених двійкових векторів ознак класів розпізнавання, що дорівнюють 0,5 для усіх класів розпізнавання та значення параметра поля контрольних допусків δ на ознаки розпізнавання.

За базовий вважають клас розпізнавання X_1^o, який характеризує нормальний стан функціонування інформаційної системи і стосовно якого визначаються контрольні допуски.

До етапів реалізації базового алгоритму належать [22]:

а) обчислення для навчальної матриці класу розпізнавання X_1^o усередненого вектору ознак $\left\{y_{1,i} | \ i = \ \overline{1,N}\right\}$.

б) формування масиву $\left\{x_{1,i}^{(j)}\right\}$ двійкових векторів ознак класу розпізнавання X_1^o за правилом

$$x_1^{(j)} = \begin{cases} 1, if\ y_{1,i} - \delta \le y_{1,i}^{(j)} \ge y_{1,i} + \delta, \\ 0, if\ else. \end{cases} \quad (2.8)$$

в) формування масиву усереднених двійкових векторів-реалізацій $\{x_{m,i} | m = \overline{1,M}, i = \overline{1,N}\}$, елементи яких обраховуються за правилом

$$x_{m,i} = \begin{cases} 1, if\ \frac{1}{n} \sum_{j=1}^{n} x_{m,i}^{(j)} > \rho_m, \\ 0, if\ else, \end{cases}$$

де ρ_m – рівень селекції координат двійкового вектора $x_m \in X_m^o$.

г) розбиття множини усереднених векторів ознак за правилом «найближчих сусідів» $\Re_m^{|2|} = \langle x_m,\ x_l \rangle$, де x_l є усередненим вектором ознак сусіднього класу X_l^o здійснюється за такою схемою:

1) структурування множини векторів $\{x_m\}$, починаючи з вектора x_1 базового класу X_1^o, яка характеризує нормальний стан функціонування інформаційної системи;
2) побудова матриці кодових відстаней між усередненими векторами ознак всіх класів розпізнавання розмірністю $M \times M$;
3) обчислення мінімального елементу, що належить стовпчику вектора, для кожного ряду матриці кодових відстаней, найближчого до вектора, що визначає рядок. У разі, якщо виявлено наявність декількох однакових мінімальних елементів, то обирають будь-який з них, так як вони є рівноправними;
4) формування структурованої множини елементів попарного розбиття $\left\{\Re_m^{|2|} |\ m = \overline{1,M}\right\}$, що задає план машинного навчання.

д) оптимізація кодової відстані d_m, яка змінюється за заданим законом. У даному випадку беруть $E_m(0) = 0$.

е) закінчення процедури у випадку знаходження максимуму інформаційного критерію оптимізації параметрів машинного навчання в робочій області визначення його функції.

Тобто, базовий алгоритм навчання є ітераційною процедурою пошуку глобального максимуму інформаційного критерію оптимізації параметрів машинного навчання в робочій області визначення його функції:

$$d_m^* = \arg \max_{G_E \cap \{d\}} E_m^*(d). \quad (2.9)$$

Під час машинного навчання за базовим алгоритмом виконується покрокове збільшення радіуса d_1 контейнера класу розпізнавання X_1^o на одну кодову одиницю. Разом з тим, відповідно до умови (2.5) на величину радіуса d_1 накладається обмеження

$$d_1 < d(x_1 \oplus x_2) - 1.$$

Значення критерію E_1 оптимізації радіуса контейнера класу розпізнавання X_1^o обчислюється на кожному кроці машинного навчання. У якості оптимального радіусу обирають екстремальне значення глобального максимуму критерію E_1, що обчислюється в робочій області визначення функції інформаційного критерію. У двоальтернативних рішеннях робоча область існує в умовах, коли їх первинна та друга достовірності перевищують помилки першого та другого родів.

Отже, основною функцією базового алгоритму машинного навчання в рамках IEI-технології вважають обчислення на кожному кроці навчання

інформаційного критерію та пошук його глобального максимуму в робочій області визначення функції критерію задля обчислення оптимальних геометричних параметрів розбиття простору ознак на класи розпізнавання. У випадку гіперсферичного контейнера класів розпізнавання такими параметрами в інформаційно - екстремальному машинному навчанні за базовим алгоритмом є оптимальні усереднені вектори-реалізації $\{x_m^*\}$ для заданого алфавіту $\{X_m^o\}$ та оптимальні кодові відстані $\{d_m^*\}$ [22].

2.4 Функціонування системи розпізнавання в режимі екзамену

Після машинного навчання будуються вирішальні правила за одержаними оптимальними геометричними параметрами контейнерів класів розпізнавання. Вони подаються у вигляді [20]:

$$\left(\forall X_k^0 \in \widetilde{\Re}^{|M|}\right)\left(\forall X_l^0 \in \widetilde{\Re}^{|M|}\right)\left(\begin{array}{c} f\left[(\mu_m > 0)\,\&\,(\mu_m = \max\limits_{\{m\}}\{\mu_m\})\right] \\ then\ x^{(j)} \in X_m^o\ else\ x^{(j)} \notin X_m^o \end{array}\right), \quad (2.10)$$

де $x^{(j)}$ – вектор, який розпізнається;

μ_m – функція належності вектора $x^{(j)}$ контейнеру класу розпізнавання X_m^o.

За формулою (2.10) функція належності для гіперсферичного контейнера класу розпізнавання X_m^o визначається за формулою

$$\mu_m = 1 - \frac{d(x_m^* \oplus x^{(j)})}{d_m^*}, \quad (2.11)$$

де x_m^* є оптимальним усереднений двійковий вектор ознак;

d_m^* є оптимальним радіусом гіперсферичного контейнера.

Отже, у випадку функціонування системи виявлення атак в режимі екзамену, вирішальні правила (2.10) визначають належність реалізації класу, що

розпізнається до одного з класів із заданого алфавіту. У той же час вирішальні правила виокремлюються високою оперативністю через малу обчислювальну трудомісткість.

Функціональна ефективність інформаційно-екстремального машинного навчання оцінюється коли СВА функціонує у режимі екзамену, алгоритм якого подібний алгоритму функціонування системи в режимі роботи.

В рамках IEI-технології категорійну модель, яка застосовується в режимі екзамену, подано у вигляді орієнтованого графа відображень множин на рис. 2.2 [20].

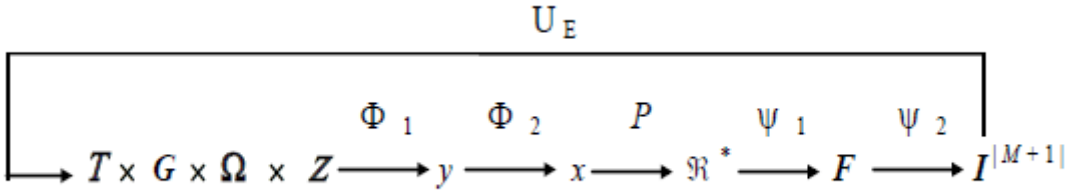


Рисунок 2.2 – Категорійна модель функціонування СВА в режимі екзамену

У категорійній моделі (рис. 2.3) оператор Φ_1 формує екзаменаційний вектор ознак класу, який розпізнається, аналогічний за структурою вектору з навчальної матриці. Оператор у Φ_2 формує двійковий вектор x за контрольними допусками, вказаними на етапі машинного навчання вказаними, у свою чергу, оператор P відображає цей вектор на оптимальне розбиття $\Re^*$ класів розпізнавання, побудоване на етапі машинного навчання. Оператор Ψ_1 для кожного вектора ознак, які розпізнаються, обчислює значення вирішальних правил, які були побудовані на етапі машинного навчання і формує терммножину F, а оператор Ψ_2 за максимальним значенням вирішального правила відносить вектор x, до одного з класів заданого алфавіту $\{X_m^o\}$. У цьому разі множина гіпотез $I^{|M+1|}$ містить додаткову гіпотезу γ_{M+1}, яка характеризує некласифіковане рішення, тобто вектор, що розпізнається, не належить до

жодного класу із заданого алфавіту класів розпізнавання. Призначенням оператора U_E є регламентація процесу екзамену [20].

Реалізація алгоритму екзамену відбувається за переліком, вказаним нижче [20]:

а) ініціалізується лічильник класів розпізнавання $m := 0$;

б) $m := m + 1$;

в) ініціалізується лічильник кількості реалізацій $j := 0$;

г) $j := j + 1$;

д) обчислюється функція належності (2.11);

е) виконується порівняння: якщо $j \leq n$, то потрібно виконати пункт г, інакше – пункт ж;

ж) обчислюється усереднене значення функції належності (2.11):

$$\overline{\mu_m} = \frac{1}{n}\sum_{j=1}^{n} \mu_{m,j}\,; \tag{2.12}$$

з) виконується порівняння: якщо $m \leq M$, то потрібно виконати пункт б, інакше – пункт и;

и) обчислюється максимальне значення функції (2.12):

$$\bar{\mu}_m^* = \max_{\{m\}} \bar{\mu}_m\,;$$

к) визначення класу розпізнавання за максимальним значенням функції (2.12);

л) у разі, якщо для всіх класів розпізнавання максимальні значення функції (2.12) від'ємні, то екзаменаційна реалізація не класифікується;

м) зупиняється алгоритм.

Таким чином, вирішальні правила, які були побудовані в рамках геометричного підходу в процесі машинного навчання є чіткими та відрізняються лише невеликою обчислювальною трудомісткістю. Тому вони характеризуються високою оперативністю, яка є досить важливим фактором під час функціонування системи виявлення атак.

3 ІНФОРМАЦІЙНЕ, АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ВИЯВЛЕННЯ АТАК

3.1 Формування вхідного математичного опису системи виявлення атак

Формуючи вхідний математичний опис інформаційно-екстремального навчання системи виявлення атак були використані дані, взяті з відкритого репозиторію даних «Machine Learning Repository» для машинного навчання [24]. Вказаний репозиторій пропонує дані реального трафіку, зібрані з функціонуючих пристроїв.

Кожен трафік містить 115 таких характеристик (табл. 3.1):

Таблица 3.1 – Характеристики трафіків

№	Характеристика	№	Характеристика	№	Характеристика
1	MI_dir_L5_weight	40	HH_L3_std	78	HH_jit_L0.01_weight
2	MI_dir_L5_mean	41	HH_L3_magnitude	79	HH_jit_L0.01_mean
3	MI_dir_L5_variance	42	HH_L3_radius	80	HH_jit_L0.01_variance
4	MI_dir_L3_weight	43	HH_L3_covariance	81	HpHp_L5_weight
5	MI_dir_L3_mean	44	HH_L3_pcc	82	HpHp_L5_mean
6	MI_dir_L3_variance	45	HH_L1_weight	83	HpHp_L5_std
7	MI_dir_L1_weight	46	HH_L1_mean	84	HpHp_L5_magnitude
8	MI_dir_L1_mean	47	HH_L1_std	85	HpHp_L5_radius
9	MI_dir_L1_variance	48	HH_L1_magnitude	86	HpHp_L5_covariance
10	MI_dir_L0.1_weight	49	HH_L1_radius	87	HpHp_L5_pcc
11	MI_dir_L0.1_mean	50	HH_L1_covariance	88	HpHp_L3_weight
12	MI_dir_L0.1_variance	51	HH_L1_pcc	89	HpHp_L3_mean

Продовження таблиці 3.1

№	Характеристика	№	Характеристика	№	Характеристика
13	MI_dir_L0.01_weight	52	HH_L0.1_weight	90	HpHp_L3_std
14	MI_dir_L0.01_mean	53	HH_L0.1_mean	91	HpHp_L3_magnitude
15	MI_dir_L0.01_variance	54	HH_L0.1_std	92	HpHp_L3_radius
16	H_L5_weight	55	HH_L0.1_magnitude	93	HpHp_L3_covariance
17	H_L5_mean	56	HH_L0.1_radius	94	HpHp_L3_pcc
18	H_L5_variance	57	HH_L0.1_covariance	95	HpHp_L1_weight
19	H_L3_weight	58	HH_L0.1_pcc	96	HpHp_L1_mean
20	H_L3_mean	59	HH_L0.01_weight	97	HpHp_L1_std
21	H_L3_variance	60	HH_L0.01_mean	98	HpHp_L1_magnitude
22	H_L1_weight	61	HH_L0.01_std	99	HpHp_L1_radius
23	H_L1_mean	62	HH_L0.01_magnitude	100	HpHp_L1_covariance
24	H_L1_variance	63	HH_L0.01_radius	101	HpHp_L1_pcc
25	H_L0.1_weight	64	HH_L0.01_covariance	102	HpHp_L0.1_weight
26	H_L0.1_mean	65	HH_L0.01_pcc	103	HpHp_L0.1_mean
27	H_L0.1_variance	66	HH_jit_L5_weight	104	HpHp_L0.1_std
28	H_L0.01_weight	67	HH_jit_L5_mean	105	HpHp_L0.1_magnitude
29	H_L0.01_mean	68	HH_jit_L5_variance	106	HpHp_L0.1_radius
30	H_L0.01_variance	69	HH_jit_L3_weight	107	HpHp_L0.1_covariance
31	HH_L5_weight	70	HH_jit_L3_mean	108	HpHp_L0.1_pcc
32	HH_L5_mean	71	HH_jit_L3_variance	109	HpHp_L0.01_weight
33	HH_L5_std	72	HH_jit_L1_weight	110	HpHp_L0.01_mean
35	HH_L5_magnitude	73	HH_jit_L1_mean	111	HpHp_L0.01_std

Продовження таблиці 3.1

№	Характеристика	№	Характеристика	№	Характеристика
35	HH_L5_radius	74	HH_jit_L1_variance	112	HpHp_L0.01_magnitude
36	HH_L5_covariance	75	HH_jit_L0.1_weight	113	HpHp_L0.01_radius
37	HH_L5_pcc	76	HH_jit_L0.1_mean	114	HpHp_L0.01_covariance
38	HH_L3_weight	77	HH_jit_L0.1_variance	115	HpHp_L0.01_pcc
39	HH_L3_mean				

Вищезазначений репозиторій надає такі описи кожного з заголовків :

- H – статистика, яка підсумовує останній трафік від хосту цього пакета (IP);
- HH – статистика, яка підсумовує нещодавній трафік, що йде від хоста цього пакета до хосту призначення пакета;
- HpHp – статистика, що підсумовує останній трафік, що йде від хосту та порту цього пакета до хосту та порту призначення пакета;
- HH_jit – статистика, яка підсумовує тремтіння трафіку, що йде від хосту цього пакета до хосту призначення пакета;
- weight – розглядається у якості кількості пакетів, які відображаються у нещодавній історії (вага потоку);
- radius – квадратний корінь суми дисперсій двох потоків;
- magnitude – квадратний корінь суми середніх значень двох потоків;
- cov – наближена коваріація між двома потоками;
- pcc – коефіцієнт кореляції між двома потоками;
- L – проміжки часу.

У дипломній роботі використано нормальний трафік та чотири види інфікованого трафіку, тому навчальна матриця складалася з 4 станів мережі, та

мала розмірність 100×115, де n = 100 і задає кількість реалізацій, а m=115 – кількість ознак.

3.2 Короткий опис програмного забезпечення

Для програмної реалізації лінійного алгоритму системи виявлення атак було використано систему MATLAB. Matrix Laboratory – це пакет прикладних програм для вирішення задач різноманітних технічних обчислень. Система розроблена американською компанією MathWorks та досить активно розвивається і по наш час. Matlab формується з досить великої кількості спеціальних програм, які вирішують цілий спектр різних технічних та математичних проблем у різноманітних галузях науки. Крім того, система включає в себе приблизно 60 різних спеціалізованих наборів інструментів («Toolboxes»), що допомагають у реалізації окремих випадків у різних сферах науки.

Для функціонування системи були створені основні функції, які описані в табл. 3.2.

Таблица 3.2 – Функції, створені під час реалізації алгоритму

Назва функції	Призначення
Input_data	Функція для зчитування даних з файлів та занесення в відповідні масиви
Bin_Matrix	Функція для формування бінарної навчальної матриці
Etalon_vector	Функція для обчислення усереднених векторів ознак
Near_Class	Функція для визначення найближчого класу
Code_distance_2	Функція для обчислення кількісті кодових відстаней від геометричних центрів контейнерів класів до їх реалізацій

Продовження таблиці 3.2

Назва функції	Призначення
Precise_characteristics	Функція для визначення точнісних характеристик
OR_Shenon	Функція обчислення оптимального радіусу за критерієм Шенона
OR_Kulbaka	Функція обчислення оптимального радіусу за критерієм Кульбака
Exam_algorithm	Функція для реалізації алгоритму екзамену

У таблиці 3.3 наведені змінні, які використовувалися під час реалізації лінійного алгоритму інформаційно- екстремального машинного навчання.

Таблиця 3.3 – Змінні, використані під час реалізації алгоритму

Змінна	Опис
X	Вхідний трафік
Y	Навчальна матриця
BinMatr	Бінарна навчальна матриця
EV	Усереднений вектор ознак
near_BM_	Найближчий клас до заданого
SK	Масив кодових відстаней
K1	Кількість своїх реалізацій
K2	Кількість чужих реалізацій
D1	Перша достовірність, або імовірність того, що свої реалізації будуть розпізнані як свої
D2	Друга достовірність, або імовірність того, що чужі реалізації будуть розпізнані як чужі

Продовження таблиці 3.3

Змінна	Опис
alfa	Помилка першого роду, або імовірність того, що свої реалізації не будуть розпізнані як свої
beta	Помилка другого роду, або імовірність того, що чужі реалізації будуть розпізнані як свої
E	Ентропійна міра Шеннона
E_max	Максимальне значення ентропійної міри Шеннона
Rab_obl	Робоча область
del_opt	Оптимальне значення δ
VD	Верхній допуск
ND	Нижній допуск
J	Інформаційна міра Кульбака

У таблиці 3.4 наведені основні змінні, які використовувалися під час реалізації алгоритму екзамену.

Таблиця 3.4 – Змінні, використані під час реалізації алгоритму екзамену

Змінна	Опис
EXAM	Екзаменаційна матриця
BinMatr_EXAM	Бінарна екзаменаційна матриця
SK_EXAM	Кодова відстань від екзаменаційної матриці до усереднених векторів ознак
mu	Значення функції належності до класу розпізнавання

3.3 Результати моделювання

Під час реалізації лінійного алгоритму інформаційно-екстремального машинного навчання СВА було використано 4 класи: X_1^0 – нормальний трафік, X_2^0, X_3^0, X_4^0 – заражені трафіки. За базовий клас було обрано X_1^0, відносно нього були визначені і контрольні допуски. З метою пошуку оптимальних значень параметрів було проведено паралельну оптимізацію контрольних допусків. У результаті чого було отримано задану на рис. 3.2 робочу область зі вказаним значенням глобального максимуму критерію E за ентропійною мірою Шеннона.

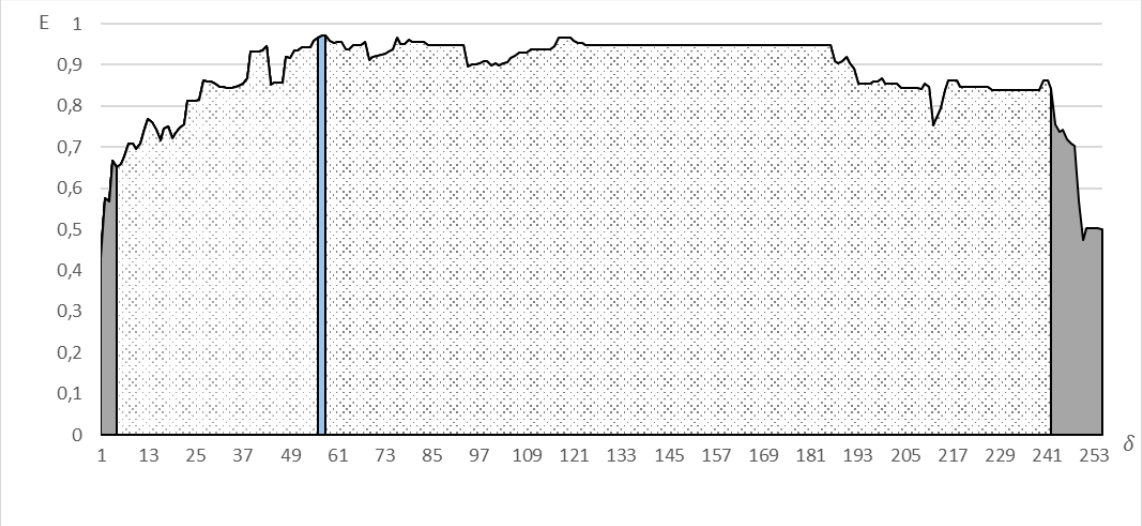


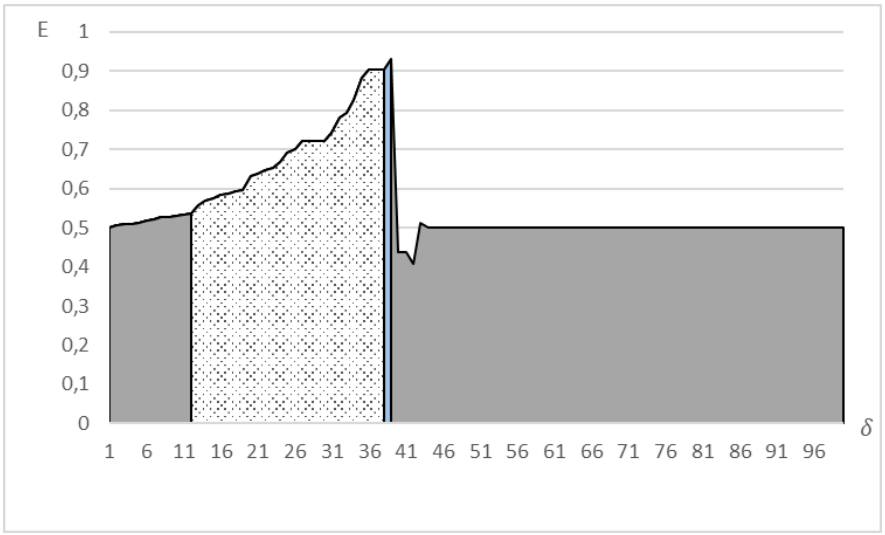
Рисунок 3.1 – Графік залежності інформаційного критерію від δ

З графіку видно, що максимальне значення критерію E було досягнуто на 57 та 58 кроках і складає $E = 0.9726$, тобто оптимальне значення параметру для контрольних допусків становить $\delta = 57$.

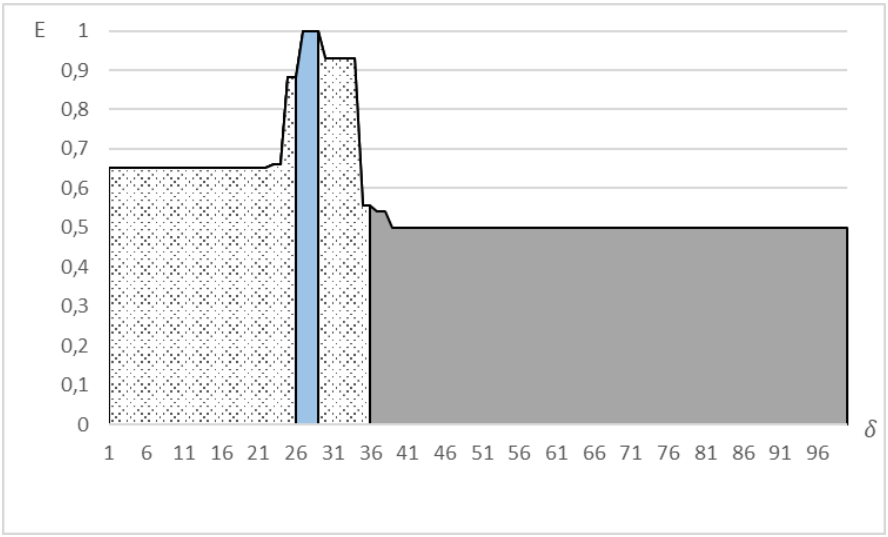
Застосовуючи оптимальний крок δ було побудовано графіки для кожного класу за критеріями Шеннона та Кульбака, які демонструють залежність інформаційного критерію від радіусів контейнерів класу.

На рис. 3.2 подані відображення результатів оптимізації геометричних параметрів за критерієм Шеннона:

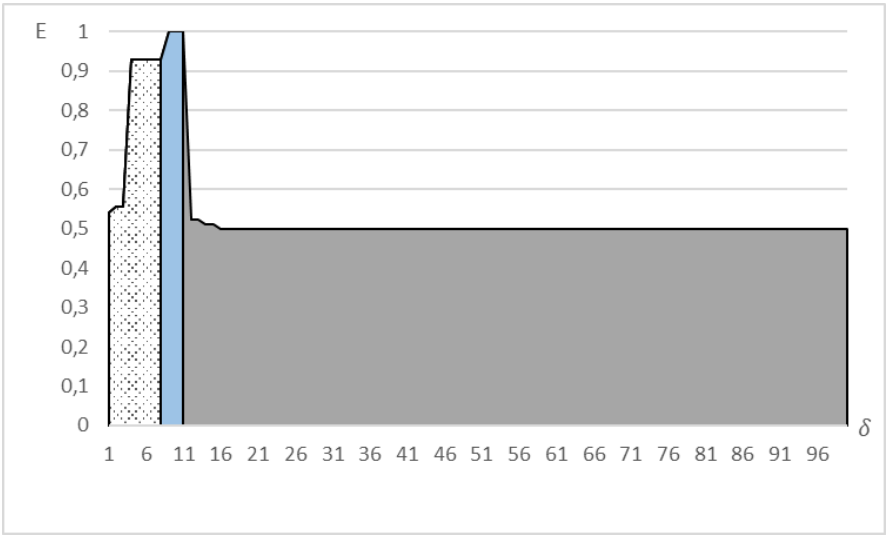
а)


E
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
δ
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96

б)


E
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
δ
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96

в)


E
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
δ
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96

г)

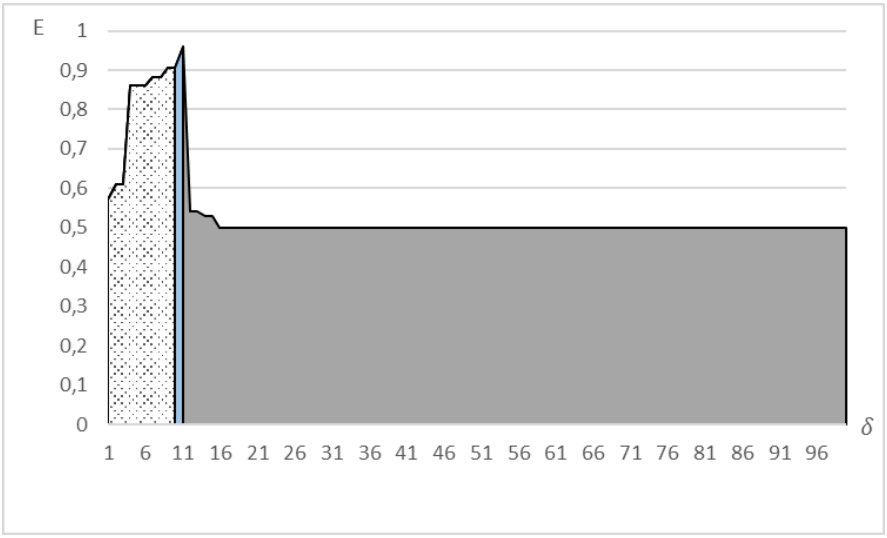


Рисунок 3.2 – Графіки залежності інформаційного критерію від радіусів за критерієм Шеннона для класів: а) X_1^0, б) X_2^0, в) X_3^0, г) X_4^0

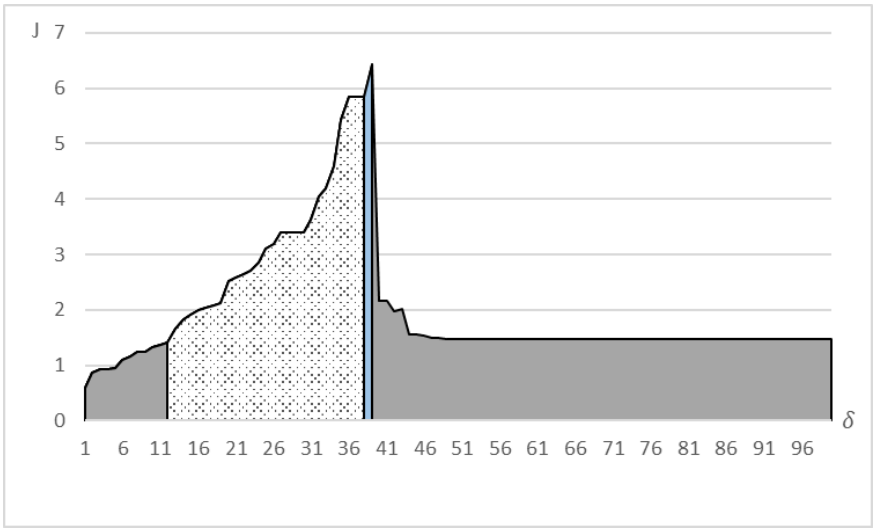
Результати після аналізу кожного з графіків наведені в таблиці 3.5:

Таблиця 3.5 – Результати

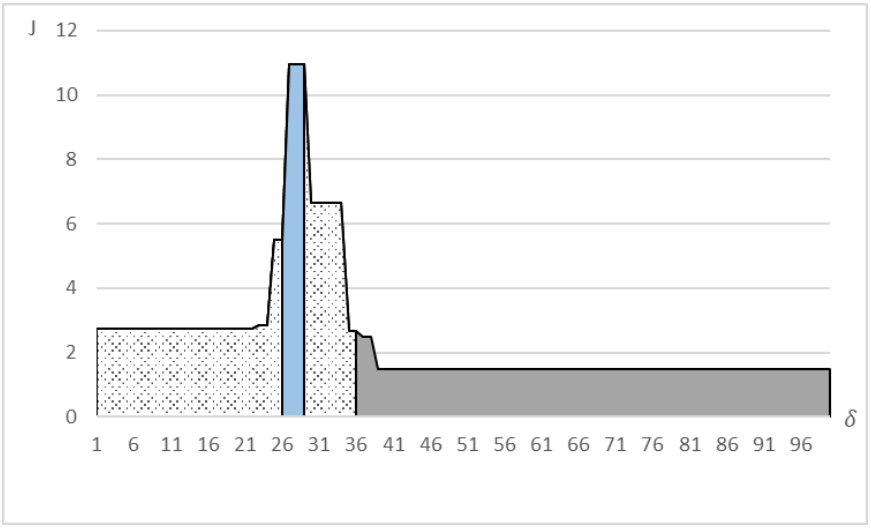
Клас	E	δ
X_1^0	0.9304	39
X_2^0	1	27
X_3^0	1	9
X_4^0	0.9599	11

На рис. 3.3 подані аналогічні відображення, тільки за критерієм Кульбака:

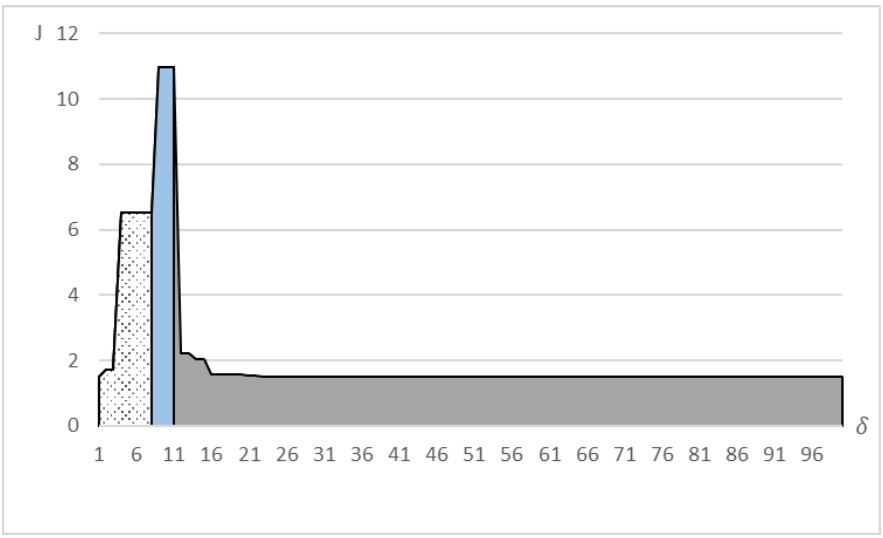
а)



б)



в)



г)

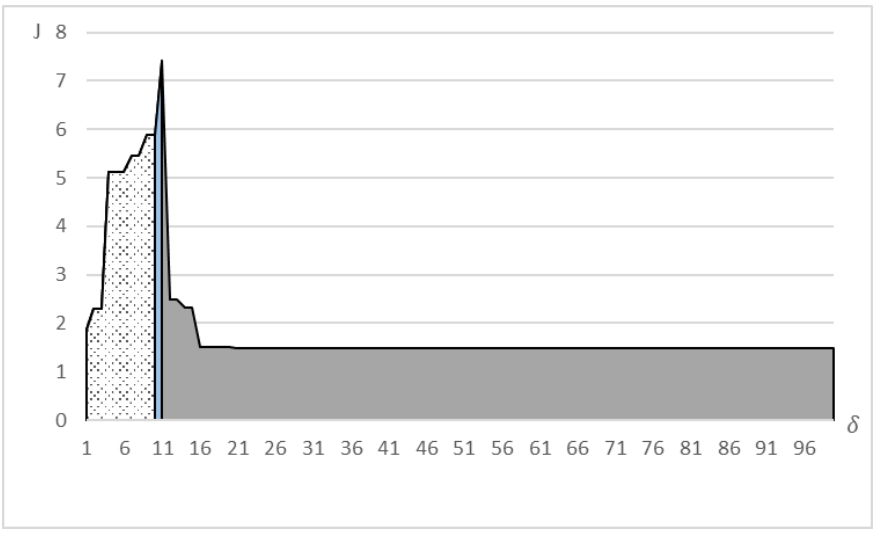


Рисунок 3.3 – Графіки залежності інформаційного критерію від радіусів за критерієм Кульбака для класів: а) X_1^0, б) X_2^0, в) X_3^0, г) X_4^0

Результати після аналізу кожного з графіків наведені в таблиці 3.6:

Таблиця 3.6 – Результати

Клас	J	δ
X_1^0	6.4285	39
X_2^0	10.9665	27
X_3^0	10.9665	9
X_4^0	7.4248	11

Аналізуючи дані, отримані за обома критеріями при оптимальному параметрі контрольних допусків, відзначимо, що радіуси контейнерів мають однакові значення.

Під час реалізації алгоритму екзамену сформовано екзаменаційну матрицю, яка містить дані чотирьох трафіків, досліджених у даній роботі, у такому співвідношенні: $X1(1{:}50,:)$, $X2(51{:}70,:)$, $X3(91{:}100,:)$ та $X4(71{:}90,:)$.

Після проведення алгоритму екзамену були отримані такі значення $\mu_1 = 0.1005$, $\mu_2 = -0.7822$, $\mu_3 = -3.1778$ та $\mu_4 = -1.9818$ функцій належності класам X_1^0, X_2^0, X_3^0, X_4^0 відповідно. У результаті цього система прийняла рішення про належність екзаменаційної реалізації до класу X_1^0, що відповідає дійсності.

Отже, можна зробити висновок, що машинне навчання пройшло успішно та система працює відповідно заданим вимогам.

ВИСНОВКИ

1. У випускній роботі проведений детальний аналіз систем виявлення атак. Проаналізовано найпопулярніші системи, які позиціонують себе як системи виявлення атак. Досліджено загальні правила класифікації СВА. Розглянуто переваги та недоліки двох рівнів СВА, до яких відносять мережевий та на базі хосту. Зроблено висновок, що задля забезпечення повного захисту потрібне гібридне використання одночасно двох видів СВА. Було розглянуто два основні методи виявлення атак, до яких відносять сигнатурний метод і метод виявлення аномалій та проведено порівняння даних методів. Дослідження показало, що в даний час зріс інтерес до створення комбінованих СВА, які можуть поєднувати у собі обидва методи виявлення атак.

2. У процесі реалізації інформаційної технології функціонування системи виявлення атак було застосовано інформаційно-екстремальну інтелектуальну технологію аналізу даних. Під час інформаційного синтезу даної системи розглянуто категорійну модель інформаційно-екстремального машинного навчання, що виступає узагальненою архітектурою базового алгоритму машинного навчання. Після цього проведено формування вхідного математичного опису та визначення відповідного базового класу розпізнання. Формуючи вхідний математичний опис інформаційно-екстремального навчання системи виявлення атак були використані дані, взяті з відкритого репозиторію даних.

3. Розробка проведена за допомогою MATLAB – пакету прикладних програм для вирішення завдань технічних обчислень. На етапі навчання було обчислено оптимальні геометричні параметрами контейнерів класів розпізнавання. На етапі реалізації алгоритму екзамену обчислено функцію належності для гіперсферичного контейнера класу розпізнавання. Тобто визначено належність реалізацій класу, що розпізнається до відповідного класу із заданого алгоритму.

СПИСОК ЛІТЕРАТУРИ

1. Закон України Про основні засади забезпечення кібербезпеки України [Електронний ресурс]. – Режим доступу : https://zakon.rada.gov.ua/laws/show/2163-19#Text.
2. ЗВІТ Національної поліції України про результати роботи у 2020 році. [Електронний ресурс]. – Режим доступу : https://www.kmu.gov.ua/storage/app/sites/1/17-civik-2018/zvit2020/npu-zvit-2020.pdf
3. Система виявлення вторгнень. [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B2%D0%B8%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D0%B2%D1%82%D0%BE%D1%80%D0%B3%D0%BD%D0%B5%D0%BD%D1%8C
4. Сєвєрінов О.В. Аналіз сучасних систем виявлення вторгнень / О.В. Сєвєрінов, А.Г. Хрєнов // Системи обробки інформації. – 2013. – Вип. 6(122). – С. 122 – 124.
5. Павлов І.М. Аналіз таксономії систем виявлення атак у контексті сучасного рівня розвитку інформаційних систем / І.М. Павлов, С.В. Толюпа, В.І. Ніщенко // Сучасний захист інформації. – 2014. – Вип. 4. – С.44 – 52.
6. NIDS – Network based intrusion detection. [Electronic resource]. – Regime of access:https://www.redscan.com/services/managed-intrusion-detection-system/nids/
7. Хостовая система обнаружения вторжений. [Електронний ресурс]. – Режим доступу : https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BE%D0%B1%D0%BD%D0%B0%D1%80%

D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D1%82%D0%BE%D1%80%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9

8. Система обнаружения вторжений на базе IDS Snort (snort ids). [Електронний ресурс]. – Режим доступу : https://www.opennet.ru/base/sec/snort_ids.txt.html
9. Установка и настройка утилиты для обнаружения вторжений в сети – Snort. [Електронний ресурс]. – Режим доступу : https://wiki.merionet.ru/seti/31/ustanovka-i-nastrojka-utility-dlja-obnaruzhenija-vtorzhenij-v-seti-snort/
10. Intrusion Detection Systems Explained: 13 Best IDS Software Tools Reviewed. [Electronic resource]. – Regime of access: https://www.comparitech.com/net-admin/network-intrusion-detection-tools/
11. Корченко А. О. Методи ідентифікації аномальних станів для систем виявлення вторгнень: монографія / А. О. Корченко; Київ. Компринт. – 2019.
12. OSSEC.Host Intrusion Detection for Everyone. [Electronic resource]. – Regime of access: https://www.ossec.net/about/
13. Установка и настройка OSSEC-HIDS. [Електронний ресурс]. – Режим доступу : https://www.tux.in.ua/articles/1810
14. Что такое Suricata. [Електронний ресурс]. – Режим доступу : https://www.dmosk.ru/terminus.php?object=suricata
15. Best FREE Intrusion Detection Software in 2021. [Electronic resource]. – Regime of access: https://www.addictivetips.com/net-admin/intrusion-detection-tools/
16. Осваимаем сетевую IDS/IPS Suricata. [Електронний ресурс]. – Режим доступу : https://xakep.ru/2015/06/28/suricata-ids-ips-197/
17. Zeek. [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Zeek

18. Top 10 BEST Intrusion Detection Systems (IDS) [2021 Rankings]. [Electronic resource]. – Regime of access: https://www.softwaretestinghelp.com/intrusion-detection-systems/
19. Зоріна Т.І. Система виявлення і запобігання атак в комп'ютерних мережах. / Т. І. Зоріна // Вісник Східноукраїнського національного університету імені Володимира Даля . – 2013. – Вип. 15 (204). – С. 48 – 52.
20. Сучасні інформаційні технології в кібербезпеці : монографія / [А. С. Довбиш, В. К. Ободяк, І. В. Шелехов та ін.]; за ред. В. К. Ободяка, І. В. Шелехова. – Суми : Сумський державний університет, 2021. – 348 с.
21. Довбиш, А.С. Інформаційно-екстремальний алгоритм машинного навчання системи розпізнавання транспортних засобів. / А. С. Довбиш, Ю. В. Симоновський, О. В. Коробченко, М. А. Летюга // Вісник НТУ «ХПІ». – 2016. – Вип. 45 (1217). – С. 22-28.
22. Довбиш, А.С. Основи проектування інтелектуальних систем : навч. посіб. / А.С. Довбиш; Суми. Сумський Державний Університет. – Суми : СумДУ, 2009. – 170 с.
23. Довбиш, А.С. Оптимізація ієрархічної структури даних інтелектуальної системи функціонального діагностування технічного стану складної машини. / А. С. Довбиш, В.І. Зимовець, М. В. Бібик // Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – 2018. – Вип. 44 (1320). – С.42- 49.
24. Machine Learning Repository. [Electronic resource]. – Regime of access: https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.

ДОДАТОК А

Опис основного коду:

```
file1=fopen('N.txt');
file2=fopen('B1.txt');
file3=fopen('B2.txt');
file4=fopen('B3.txt');
[f1,X1] = Input_data( file1 );
[f2,X2] = Input_data( file2 );
[f3,X3] = Input_data( file3 );
[f4,X4] = Input_data( file4 );
[X1, X2, X3, X4] = Data_normalization(X1, X2, X3, X4);
Y(:,:,1)=X1;
Y(:,:,2)=X2;
Y(:,:,3)=X3;
Y(:,:,4)=X4;

[n m k]=size(Y);

mean_tr1=mean(Y(:,:,1));

for del=1:255
VD=mean_tr1+del;
ND=mean_tr1-del;

 [BinMatr1,BinMatr2,BinMatr3,BinMatr4] = Bin_Matrix(Y,ND,VD,n);

 [EV1,EV2,EV3,EV4] = Etalon_vector(BinMatr1,BinMatr2,BinMatr3,BinMatr4);

 [near_BM_1, dc_1] = Near_Class( EV1, EV2, EV3, EV4, BinMatr2, BinMatr3,
BinMatr4);
[near_BM_2, dc_2] = Near_Class( EV2, EV1, EV3, EV4, BinMatr1, BinMatr3,
BinMatr4);
[near_BM_3, dc_3] = Near_Class( EV3, EV1, EV2, EV4, BinMatr1, BinMatr2,
BinMatr4);
[near_BM_4, dc_4] = Near_Class( EV4, EV1, EV2, EV3, BinMatr1, BinMatr2,
BinMatr3);
```

```
 [SK1, SK_from_1_to_near] = Code_distance_2(EV1,BinMatr1,near_BM_1,n);
[SK2, SK_from_2_to_near] = Code_distance_2(EV2,BinMatr2,near_BM_2,n);
[SK3, SK_from_3_to_near] = Code_distance_2(EV3,BinMatr3,near_BM_3,n);
[SK4, SK_from_4_to_near] = Code_distance_2(EV4,BinMatr4,near_BM_4,n);

radius=1:n;
 [K1_1, K2_1, D1_1, D2_1, alfa1, beta1] =
Precise_characteristics(SK1,SK_from_1_to_near,radius,n);
[K1_2, K2_2, D1_2, D2_2, alfa2, beta2] =
Precise_characteristics(SK2,SK_from_2_to_near,radius,n);
[K1_3, K2_3, D1_3, D2_3, alfa3, beta3] =
Precise_characteristics(SK3,SK_from_3_to_near,radius,n);
[K1_4, K2_4, D1_4, D2_4, alfa4, beta4] =
Precise_characteristics(SK4,SK_from_4_to_near,radius,n);

 [E1, E_max1, tochn_char1, rab_obl1, optim_radius_E1, D1_1, D2_1 ] =
OR_Shenon( alfa1, beta1, D1_1, D2_1, radius, K1_1, K2_1, dc_1);
[E2, E_max2, tochn_char2, rab_obl2, optim_radius_E2, D1_2, D2_2 ] =
OR_Shenon( alfa2, beta2, D1_2, D2_2, radius, K1_2, K2_2, dc_2);
[E3, E_max3, tochn_char3, rab_obl3, optim_radius_E3, D1_3, D2_3 ] =
OR_Shenon( alfa3, beta3, D1_3, D2_3, radius, K1_3, K2_3, dc_3);
[E4, E_max4, tochn_char4, rab_obl4, optim_radius_E4, D1_4, D2_4 ] =
OR_Shenon( alfa4, beta4, D1_4, D2_4, radius, K1_4, K2_4, dc_4);

Edel(del)= (E_max1+E_max2+E_max3+E_max4)./4;
if((D1_1>0.5)&(D2_1>0.5)&(D1_2>0.5)&(D2_2>0.5)&(D1_3>0.5)&(D2_3>0.5)&(
D1_4>0.5)&(D2_4>0.5) )
   Rab_obl(del)=del; end;
end

Rab_obl=find(Rab_obl>0)

Edel_max=max(Edel(Rab_obl))
del_opt=Rab_obl(find(Edel(Rab_obl)==Edel_max))
del_opt=del_opt(1)

VD=mean_tr1+del_opt;
```

```
ND=mean_tr1-del_opt;

 [BinMatr1,BinMatr2,BinMatr3,BinMatr4] = Bin_Matrix(Y,ND,VD,n);

 [EV1,EV2,EV3,EV4] = Etalon_vector(BinMatr1,BinMatr2,BinMatr3,BinMatr4);

[near_BM_1, dc_1] = Near_Class( EV1, EV2, EV3, EV4, BinMatr2, BinMatr3,
BinMatr4);
[near_BM_2, dc_2] = Near_Class( EV2, EV1, EV3, EV4, BinMatr1, BinMatr3,
BinMatr4);
[near_BM_3, dc_3] = Near_Class( EV3, EV1, EV2, EV4, BinMatr1, BinMatr2,
BinMatr4);
[near_BM_4, dc_4] = Near_Class( EV4, EV1, EV2, EV3, BinMatr1, BinMatr2,
BinMatr3);

[SK1, SK_from_1_to_near] = Code_distance_2(EV1,BinMatr1,near_BM_1,n);
[SK2, SK_from_2_to_near] = Code_distance_2(EV2,BinMatr2,near_BM_2,n);
[SK3, SK_from_3_to_near] = Code_distance_2(EV3,BinMatr3,near_BM_3,n);
[SK4, SK_from_4_to_near] = Code_distance_2(EV4,BinMatr4,near_BM_4,n);

radius=1:n;

[K1_1, K2_1, D1_1, D2_1, alfa1, beta1] =
Precise_characteristics(SK1,SK_from_1_to_near,radius,n);
[K1_2, K2_2, D1_2, D2_2, alfa2, beta2] =
Precise_characteristics(SK2,SK_from_2_to_near,radius,n);
[K1_3, K2_3, D1_3, D2_3, alfa3, beta3] =
Precise_characteristics(SK3,SK_from_3_to_near,radius,n);
[K1_4, K2_4, D1_4, D2_4, alfa4, beta4] =
Precise_characteristics(SK4,SK_from_4_to_near,radius,n);

[E1, E_max1, tochn_char1, rab_obl1, optim_radius_E1, D1_1, D2_1 ] =
OR_Shenon( alfa1, beta1, D1_1, D2_1, radius, K1_1, K2_1, dc_1);
[E2, E_max2, tochn_char2, rab_obl2, optim_radius_E2, D1_2, D2_2 ] =
OR_Shenon( alfa2, beta2, D1_2, D2_2, radius, K1_2, K2_2, dc_2);
[E3, E_max3, tochn_char3, rab_obl3, optim_radius_E3, D1_3, D2_3 ] =
OR_Shenon( alfa3, beta3, D1_3, D2_3, radius, K1_3, K2_3, dc_3);
```

```
[E4, E_max4, tochn_char4, rab_obl4, optim_radius_E4, D1_4, D2_4 ] =
OR_Shenon( alfa4, beta4, D1_4, D2_4, radius, K1_4, K2_4, dc_4);

[J1, optim_radius_J1, J_max1] = OR_Kulbaka(D1_1, D2_2, alfa1, beta1, rab_obl1);
[J2, optim_radius_J2, J_max2] = OR_Kulbaka(D1_2, D2_2, alfa2, beta2, rab_obl2);
[J3, optim_radius_J3, J_max3] = OR_Kulbaka(D1_3, D2_3, alfa3, beta3, rab_obl3);
[J4, optim_radius_J4, J_max4] = OR_Kulbaka(D1_4, D2_4, alfa4, beta4, rab_obl4);
```

Опис функції Input_data:

```
function [ f,X ] = Input_data( file )
f=[];
for i=1:115
   f=[f '%g'];
end;
 X=fscanf(file,f,[115 100]);
 fclose(file);
end
```

Опис функції Data_normalization:

```
function [X1, X2, X3, X4] = Data_normalization(x1, x2, x3, x4)
for i=1:115
   ximin1=(min(x1(i,:)));
   ximax1=(max(x1(i,:)));
   ximin2=(min(x2(i,:)));
   ximax2=(max(x2(i,:)));
   ximin3=(min(x3(i,:)));
   ximax3=(max(x3(i,:)));
   ximin4=(min(x4(i,:)));
   ximax4=(max(x4(i,:)));

   ximin5=min(ximin1,ximin2);
   ximin6=min(ximin5,ximin3);
   ximin=min(ximin6,ximin4);

   ximax7=max(ximax1,ximax2);
   ximax8=max(ximax7,ximax3);
   ximax=max(ximax8,ximax4);
```

```
   X1(i,:)=(x1(i,:)-ximin).*(255./(ximax-ximin));
   X2(i,:)=(x2(i,:)-ximin).*(255./(ximax-ximin));
   X3(i,:)=(x3(i,:)-ximin).*(255./(ximax-ximin));
   X4(i,:)=(x4(i,:)-ximin).*(255./(ximax-ximin));
end;
end
```

Опис функції Bin_Matrix:

```
function [BinMatr1,BinMatr2,BinMatr3,BinMatr4] = Bin_Matrix(Y,ND,VD,n)
for i=1:n
   BinMatr1(i,:)=Y(i,:,1)>=ND&Y(i,:,1)<=VD;
   BinMatr2(i,:)=Y(i,:,2)>=ND&Y(i,:,2)<=VD;
   BinMatr3(i,:)=Y(i,:,3)>=ND&Y(i,:,3)<=VD;
   BinMatr4(i,:)=Y(i,:,4)>=ND&Y(i,:,4)<=VD;
end;
end
```

Опис функції Bin_Matrix:

```
function [EV1,EV2,EV3,EV4] =
Etalon_vector(BinMatr1,BinMatr2,BinMatr3,BinMatr4)
ro=0.5;
EV1=mean(BinMatr1)>= ro;
EV2=mean(BinMatr2)>= ro;
EV3=mean(BinMatr3)>= ro;
EV4=mean(BinMatr4)>= ro;
end
```

Опис функції Near_Class:

```
function [near_class, dc] = Near_Class( EV1, EV2, EV3, EV4, BinMatr1, BinMatr2,
BinMatr3)
dc1 = sum(xor(EV1, EV2));
dc2 = sum(xor(EV1, EV3));
dc3 = sum(xor(EV1, EV4));

mas = [dc1 dc2 dc3];
for i=1:length(mas)
   if min(mas)== dc1
```

```
        near_class = BinMatr1;
        dc = dc1;
    else if min(mas)== dc2
         near_class = BinMatr2;
         dc = dc2;
        else if min(mas)== dc3
            near_class = BinMatr3;
              dc = dc3;
            end;
        end;
    end;
end;
end
```

Опис функції Code_distance_2:

```
function [SK, SK_from_to_near] = Code_distance_2(EV1,BinMatr1,BinMatr2,n)
for i=1:n
    SK(i)=sum(abs(EV1-BinMatr1(i,:)));
    SK_from_to_near(i)=sum(abs(EV1-BinMatr2(i,:)));
end;

SK=SK';
SK_from_to_near=SK_from_to_near';
end
```

Опис функції Precise_characteristics:

```
function [K1, K2, D1, D2, alfa, beta] = Precise_characteristics(SK1,SK2,radius,n)
for i=1:n
K1(i)=sum(SK1<=radius(i));
K2(i)=sum(SK2<=radius(i));
end

D1=K1/n;
alfa=1-D1;
beta=K2/n;
D2=1-beta;
end
```

Опис функції OR_Shenon:

```
function [ E, E_max, tochn_char, rab_obl, optim_radius, D1, D2 ] = OR_Shenon(
alfa, beta, D1, D2, radius, K1, K2, dc )
warning off;
e1=alfa./(alfa+D2);
e1=e1.*log2(e1);
e1(find(isnan(e1)))=0;

e2=D1./(D1+beta);
e2=e2.*log2(e2);
e2(find(isnan(e2)))=0;

e3=beta./(D1+beta);
e3=e3.*log2(e3);
e3(find(isnan(e3)))=0;

e4=D2./(alfa+D2);
e4=e4.*log2(e4);
e4(find(isnan(e4)))=0;

E=1+0.5*(e1+e2+e3+e4);

tochn_char=[radius;K1;K2;D1;D2;alfa;beta;E]';


rab_obl=find((D1>0.5)&(D2>0.5)&(radius<=dc));
if isempty(rab_obl)
   E_max=max(E);
   tmp=find(E==E_max);
   optim_radius=-1;
   D1=D1(tmp(1));
   D2=D2(tmp(1));
else E_max = max(E(rab_obl));
   optim_radius=(find(E(rab_obl)==E_max));
   optim_radius=rab_obl(optim_radius(1));
   D1=D1(optim_radius);
   D2=D2(optim_radius);
```

```
end

end
```

Опис функції OR_Kulbaka:

```
function [ J, optim_radius, J_max ] = OR_Kulbaka(D1, D2, alfa, beta, rab_obl)
J=0.5*log2((D1+D2+0.001)./(alfa + beta+0.001)).*((D1+D2)-(alfa-beta));
J_max=max(J(rab_obl));
 optim_radius=rab_obl(find(J(rab_obl)==J_max));
 optim_radius=optim_radius(1);
end
```

Опис алгоритму екзамену:

```
exam1=X1(1:50,:);
exam2=X2(51:70,:);
exam3=X3(91:100,:);
exam4=X4(71:90,:);
EXAM=[exam1;exam2;exam3;exam4];
[BinMatr_EXAM, SK_EXAM1, SK_EXAM2, SK_EXAM3, SK_EXAM4, mu1 ,
mu2, mu3, mu4] = Exam_algorithm( EXAM, EV1, EV2, EV3, EV4, ND, VD,
optim_radius_E1, optim_radius_E2, optim_radius_E3, optim_radius_E4,n);
```

Опис функції Exam_algorithm:

```
function [ BinMatr_EXAM, SK_EXAM1, SK_EXAM2, SK_EXAM3, SK_EXAM4,
mu1 , mu2, mu3, mu4] = Exam_algorithm( EXAM, EV1, EV2, EV3, EV4, ND, VD,
optim_radius_E1, optim_radius_E2, optim_radius_E3, optim_radius_E4, n)


for i=1:n
   BinMatr_EXAM(i,:)=EXAM(i,:)>=ND&EXAM(i,:)<=VD;
end;

for i=1:n
   SK_EXAM1(i)=sum(abs(EV1-BinMatr_EXAM(i,:)));
   SK_EXAM2(i)=sum(abs(EV2-BinMatr_EXAM(i,:)));
```

```
   SK_EXAM3(i)=sum(abs(EV3-BinMatr_EXAM(i,:)));
   SK_EXAM4(i)=sum(abs(EV4-BinMatr_EXAM(i,:)));
end;

mu1=1-(SK_EXAM1./optim_radius_E1);
mu2=1-(SK_EXAM2./optim_radius_E2);
mu3=1-(SK_EXAM3./optim_radius_E3);
mu4=1-(SK_EXAM4./optim_radius_E4);

mu1=mean(mu1);
mu2=mean(mu2);
mu3=mean(mu3);
mu4=mean(mu4);

('У результаті перевірки ефективності системи в режимі екзамена можна
зробити такий вивід:')
if(mu1>0)&(mu2<0)&(mu3<0)&(mu4<0) ('Принадлежність реалізації до першого
класу')
elseif(mu1<0)&(mu2>0)&(mu3<0)&(mu4<0) ('Принадлежність реалізації до
другого класу')
   elseif(mu1<0)&(mu2<0)&(mu3>0)&(mu4<0)('Принадлежність реалізації до
третього класу')
      elseif(mu1<0)&(mu2<0)&(mu3<0)&(mu4>0)('Принадлежність реалізації до
четвертого класу')
         elseif(mu1<0)&(mu2<0)&(mu3<0)&(mu4<0)('Не належить ні до одного з
класів')


elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu1>mu2)&(mu1>mu3)&(mu1>mu
4)('Принадлежність реалізації до першого класу')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu2>mu1)&(mu2>mu3)&(mu2>mu
4)('Принадлежність реалізації до другого класу')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu3>mu1)&(mu3>mu2)&(mu3>mu
4)('Принадлежність реалізації до третього класу')
```

```
elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu4>mu1)&(mu4>mu2)&(mu4>mu
3)('Принадлежність реалізації до четвертого класу')


elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu2==mu1)('Принадлежність
реалізації до першого та другого класу одночасно')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu2==mu3)('Принадлежність
реалізації другому та третєму класу одночасно')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu3==mu1)('Принадлежність
реалізації до першого та третього класу одночасно')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu4==mu1)('Принадлежність
реалізації до першого та четвертого класу одночасно')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu4==mu2)('Принадлежність
реалізації другому та четвертому класу одночасно')

elseif(mu1>0)&(mu2>0)&(mu3>0)&(mu4>0)&(mu4==mu3)('Принадлежність
реалізації третьому і четвертому класу одночасно')

end
end
```