Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи виявлення мережних аномалій з використанням штучних нейронних мереж

(тема)

Виконав:

студент II курсу, групи КСМм-21-1

Блохін О.О.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Міхаль О.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Блохіну Олексію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи виявлення мережних аномалій з використанням штучних нейронних мереж

затверджена наказом по університету від “ 07 ” листопада 2022 р. № 1453 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2022 р.

3. Вхідні дані до роботи
штучна імунна мережа
штучна нейронна мережа
карти Кохонена

4. Перелік питань, що потрібно опрацювати у роботі
Проблеми виявлення та класифікації мережних атак
Методи обчислювального інтелекту в СВА
Програмна реалізація СВА

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання. Аналіз літератури.	08.11.2022–15.11.2022	
2	Огляд існуючих алгоритмів та методів.	16.11.2022–22.11.2022	
3	Аналіз існуючих архітектур.	23.11.2022–28.11.2022	
4	Реалізація розглянутих методів.	29.11.2022–02.12.2022	
5	Вибір даних для тестування.	03.12.2022–04.12.2022	
6	Отримання результатів	05.12.2022–06.12.2022	
7	Оформлення ПЗ	07.12.2022–12.12.2022	

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Міхаль О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 62 с., 11 рис., 1 дод., 23 джерела.

ШТУЧНА НЕЙРОННА МЕРЕЖА, ІМУННА СИСТЕМА, КАРТА КОХОНЕНА, АЛГОРИТМ, СИСТЕМА ВИЯВЛЕННЯ АТАК.

Метою кваліфікаційної роботи є аналіз методів виявлення мережних аномалій з використанням штучних нейронних мереж та імунних мереж.

У ході виконання кваліфікаційної роботи проведено порівняльний аналіз сигнатурних та евристичних методів виявлення мережевих атак, розроблена модель штучної імунної мережі. Також проведено аналіз алгоритмів генетико-конкурентного навчання мережі Кохонена для виявлення аномальних мережевих з'єднань, розроблено програмні інструменти для тестування мережевих систем виявлення атак проведена та оцінка їх можливостей.

.

ABSTRACT

Master’s thesis: 62 pages, 11 figures, 1 appendices, 23 sources.

ARTIFICIAL NEURAL NETWORK, IMMUNE SYSTEM, KOHONEN MAP, ALGORITHM, ATTACK DETECTION SYSTEM.

The major goal of this thesis is to analyze the methods of detecting network anomalies using artificial neural networks and immune networks.

In order to a comparative analysis of signature and heuristic methods for detecting network attacks was carried out, and a model of an artificial immune network was developed. An analysis of genetic-competitive learning algorithms of the Kohonen network for detecting abnormal network connections was also carried out, software tools were developed for testing network attack detection systems, and their capabilities were evaluated.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АМВР – алгоритмічна мова високого рівня

ДКА – детерміновані кінцеві автомати

КА – кінцевий автомат

ОІ – обчислювальний інтелект

СВА – система виявлення атак

ШІ – штучний інтелект

ШІМ – штучна імунна мережа

ШНМ – штучна нейронна мережа

ВСТУП

Розробка системи виявлення атак (СВА) є одним із пріоритетних напрямків у галузі інформаційної безпеки. Важливість вирішення цього завдання обумовлюється постійним збільшенням та різноманітністю комп'ютерних мережевих загроз, реалізація яких може призводити до серйозних фінансових втрат у різних організаціях. Згідно зі статистичними даними виробників антивірусної продукції у першому кварталі 2021 р. було виявлено та відображено понад 950 млн. комп'ютерних атак, тоді як за аналогічний період 2020 р. цей показник перевищив величину 796 млн. атак. Подібне зростання атакуючих дій з кожним роком вимагає залучення істотно великих сил та тимчасових витрат з боку адміністраторів та аналітиків безпеки.

У компаніях, залучених у виробництво критично важливою продукції, для підтримки безпеки корпоративних мережевих ресурсів витрачаються великі фінансові та матеріальні засоби, спрямовані на утримання спеціального обладнання у вигляді компонентів СВА та обслуговуючого його персоналу. Для забезпечення коректної інтерпретації даних, що передаються в пакетах, необхідно виконувати їх складання в мінімальний логічний потік. Мережне з'єднання, що дозволить оперувати більш високорівневими характеристиками мережевого трафіку для виявлення аномалій, властивих мережному та транспортного рівня моделі OSI.

Для виявлення мережевих атак можуть застосовуватися як сигнатурні механізми пошуку шаблонних аномальних дій, і евристичні (статистичні, нейромережевые, імунні та інших.) підходи. У разі сигнатур рішення завдання зводиться до реалізації процедури, що виконує перевірку входження заданої байтової послідовності усередині вмісту мережних пакетів. Недоліками такого рішення є складність створення репрезентативного набору з подібними записами та обмеження у виявленні

модифікованих варіантів відомої атаки. Навпаки, евристичні підходи дозволяють виявляти приховані закономірності в аналізованих мережевих потоках. Саме ця особливість пояснює їх широку популярність у науково-дослідній спільноті та відіграє ключову роль при виборі та проектуванні ядра СОА. З іншого боку, в основі функціонування більшості комерційних та відкритих програмних рішень переважає підхід, що базується на сигнатурному зіставленні зі зразком і характеризується мінімальним числом помилкових спрацьовувань. Для збереження переваг обох підходів використовується прийом їх комбінування, який, як і раніше, залишається не повною мірою. дослідженим. Тому завдання виявлення аномальних мережевих з'єднань є актуальним, а пропонований у цьому дослідженні модельно-методичний апарат, який використовує комбінування (гібридизацію) різнорідних методів обчислювального інтелекту (ОІ) та сигнатурного аналізу, спрямований її рішення. Область ОІ охоплює дослідження біологічно інспірованих моделей (нейронних мереж, нечіткої логіки, еволюційних обчислень і т.д.), спрямованих на обробку низькорівневих даних про об'єкт без використання експертних знань. Під терміном «гібридизація» розуміється комбінування різнорідних вирішувачів на єдину систему класифікації об'єктів.

Мета кваліфікаційної роботи: аналіз методів виявлення мережних аномалій з використанням штучних нейронних мереж та імунних мереж. Об’єкт дослідження: розподілені мережеві атаки, механізми їх виявлення та розподілені системи виявлення атак.

Завдання:

- аналіз сигнатурних та евристичних методів виявлення мережевих атак;розробка моделі штучної імунної системи; аналіз алгоритмів генетико-конкурентного навчання мережі Кохонена для виявлення аномальних мережевих з'єднань; розробка програмних інструментів для тестування мережевих систем виявлення атак та оцінка їх можливостей.

1 ПРОБЛЕМИ ВИЯВЛЕННЯ ТА КЛАСИФІКАЦІЇ МЕРЕЖНИХ АТАК

1.1 Класифікація методів виявлення мережних атак

За способом інтерпретації вхідних даних методи виявлення мережних атак класифікуються на методи виявлення аномалій та методи виявлення зловживань [2]. У [3] автор припускає, що зловмисник може бути виявлений за допомогою аналізу вмісту журналу аудиту контрольованої системи та наявністю відхилень вилучених із нього записів від встановлених адміністратором. У [4] автор пропонує розглянути статистичну модель, що складається з шести компонентів: суб'єкти (користувач, процес, система), об'єкти (файли, програми, команди, пристрої), записи журналу аудиту, що є результатом дії суб'єктів над об'єктами, шаблони поведінки суб'єктів, записи, що генеруються при виявленні аномальної поведінки, та правила, що задають умови їх спрацьовування та наступні дії.

Рисунок 1.1 – Схема роботи методів виявлення зловживань

Алгоритм виявлення аномалій мережі може бути описаний наступним чином. Даними для аналізу є мережевий трафік, поданий як набір мережевих пакетів, у випадку фрагментованих лише на рівні IP. Зібрані сирі дані надалі

послужать джерелом для формування необхідної інформації для подальшого аналізу. Так, отримані дані можуть бути агреговані за певний часовий інтервал та нормалізовані з метою завдання ознакових атрибутів загального вигляду, які будуть потрібні при побудові поточного профілю активності.

Створений набір ознак порівнюється з набором характеристик нормальної діяльності об'єкта (користувача або системи) – шаблоном нормальної поведінки. Якщо спостерігається суттєва розбіжність порівнюваних параметрів, то фіксується мережева аномалія. В іншому випадку приймається рішення про те, що дані характеристики трафіку відносяться до нормальної поведінки.

Додавання та зміна шаблонів нормальної поведінки може здійснюватися як у ручному режимі, так і автоматично, причому деякі параметри, що задають налаштування поточного нормального профілю мережної активності можуть змінюватися в залежності від часу доби. Описаний алгоритм може включати кілька варіантів виконання для реалізації підсистеми перевірки на відповідність шаблону нормальної поведінки. Найпростішим із них є процедура порівняння з пороговою величиною, коли накопичені результати, що описують поточну мережну активність, порівнюються з експертно заданою числовою планкою.

У цьому підході випадок перевищення значень аналізованих параметрів зазначеної межі є ознакою мережевої аномалії. Інші підходи, включаючи цей, розглянуті докладніше. Варто зазначити, що побудова шаблону нормальної поведінки є трудомістким завданням і часто не завжди здійсненним. Так, на практиці виявляється, що не кожна аномальна поведінка є атакою. Наприклад, адміністратор мережі може застосовувати налагоджувальні утиліти, такі як ping, traceroute, mtr для діагностики мережевого оточення. Дії такого роду не переслідують якихось нелегальних умислів, проте системи виявлення аномалій розпізнають цю діяльність як властиву нелегітимній мережній активності. Однією з класичних та фундаментальних робіт, присвячених виявленню зловживань є [5]. Для

вирішення цього завдання автори даної роботи пропонують використовувати розфарбовані мережі Петрі. Кожна комп'ютерна загроза є послідовністю кроків зловмисника, які відображаються на граф станів системи з кінцевою вершиною, представленою у вигляді мети атакуючого. Пропонований апарат розширює механізм регулярних виразів за допомогою введення двох умов між сусідніми вершинами графа – передумови та постумови, що спрацьовують відповідно до та після шаблонного зіставлення, а також доповнює формальні граматики підтримкою двостороннього переходу при зміні стану всередині системи.

Крім того, як зазначають автори, запропонований ними підхід дозволяє об'єднати умовні вирази та зіставлення за шаблоном. Розглянута розфарбована мережа Петрі має не більше однієї спрямованої дуги між кожним фіксованим станом та наступним за ним переходом, характеризується наявністю тільки одного кінцевого стану, у той час як кількість початкових станів не обмежено. Перехід у такий стан можливий лише за умови спрацьовування переходу, що виникає у ті моменти, коли всі вхідні стани цього переходу мають принаймні один маркер, та системний або заданий користувачем предикат, визначений у цьому переході приймає справжнє значення.

Перед виконанням переходу в новий стан здійснюється процес уніфікації пов'язаних із маркерами змінних і значень полів спрацював події. Після цього перевіряється істинність булевого виразу, закріпленого за цим переходом. Досягнення маркером термінального стану рівносильно виявленню атаки, що описується мережею Петрі.

Виявлення зловживань дозволяє ідентифікувати несанкціоновані дії, якщо є їх точне подання у вигляді шаблонів атак. Тут під шаблоном атаки розуміється деяка сукупність дій, що явно описують конкретну атаку (наприклад правил зіставлення або висновку), застосування яких до полів ідентифікованого об'єкта дозволяє отримати однозначну відповідь про його приналежність до цієї атаки.

Як і у схемі виявлення мережевих аномалій, при виявленні зловживань у мережі (рисунок 1.1) первинними даними для аналізу є дані мережного трафіку. Виділені атрибути мережних пакетів передаються в модуль, який виконує пошук та перевірку на відповідність вхідних даних правилам та повідомляє про наявності загрози у разі позитивного спрацьовування одного із цих правил.

Ключовою проблемою при створенні будь-якої системи виявлення зловживань є питання ефективного проектування механізму завдання правил. На практиці створення вичерпної бази правил виявлення різноманітних атак є неможливим, оскільки опис різних варіацій атакуючих дій може негативно відбиватися на продуктивності системи. Навіть несуттєві зміни в атаці призводять до неможливості її виявлення методами на основі зловживань, тому правила, що задаються повинні бути універсальними і покривати якомога більше відомих модифікацій мережевих атак. Тим самим методи виявлення зловживань є ефективним інструментом для виявлення відомих типів атак, проте їх застосовність до нових атак, а також до модифікацій відомих атак є безрезультативними.

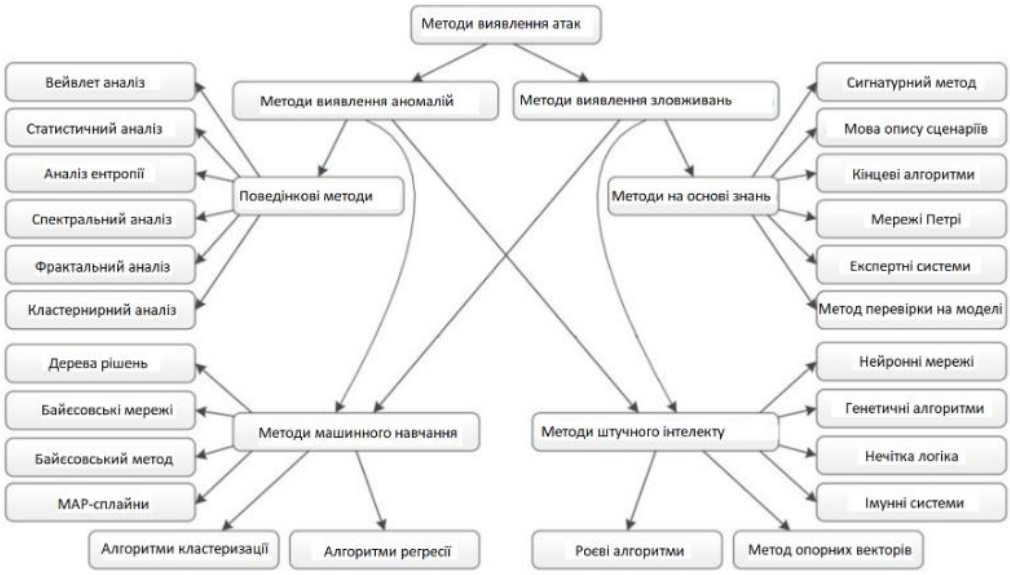

Рисунок 1.2 – Класифікація методів виявлення мережевих атак

Класифікацію методів виявлення мережевих атак, пропонована у цьому даній кваліфкаційній роботі, схематично показано на рисунку 1.2.

Ця схема розбиття є умовною і не претендує на повноту: деякі з підходів можуть належати до кількох груп. Зокрема, експертні системи та кінцеві автомати можуть використовуватися для виявлення аномалій, але в більшості випадків вони застосовуються саме для виявлення зловживань. Крім того, такі методи ОІ, як нейронні мережі та метод опорних векторів часто зараховують до методів машинного навчання. У схемі, що відповідає рисунку 1.2, обрано таке розбиття, за якого методи ОІ відокремлюються окремо завдяки біоподібній спрямованості досліджуваних у рамках ОІ алгоритмів.

Поведінковими методами називаються методи, що ґрунтуються на використання інформації про нормальну поведінку системи та її порівняння з параметрами поведінки, що спостерігається. Представлена група методів орієнтована на побудову моделі штатного, чи нормального, функціонування системи чи користувача. Системи, які використовують цей підхід, порівнюють поточні показники активності з профілем нормальної діяльності і випадок значних відхилень може розглядатися як свідчення наявності атаки.

До поведінкових методів віднесено такі методи виявлення мережевих атак:

- вейвлет-аналіз;
- статистичний аналіз;
- аналіз ентропії; спектральний аналіз;
- фрактальний аналіз;
- кластерний аналіз.

Вейвлет-аналіз полягає у побудові коефіцієнтів, що використовуються у розкладанні вихідного сигналу за базовими функціями. Як сигнал може розглядатися інтенсивність мережевого трафіку або дані про кореляцію IP-адрес призначення.

Виконання вейвлет-перетворення дозволяє виділити найвагомішу інформацію як сигнал, що відповідає коливанням з високою амплітудою і ігнорувати менш корисну інформацію в коливаннях з низькою амплітудою як шумову складову.

У іншій роботі автори як вихідні дані використовували агреговані за п'ятихвилинні інтервали середні значення наступних величин: кількість байт за секунду, кількість пакетів за секунду, кількість потоків за секунду, величину середнього розміру TCP-пакета. У кожному випадку зібрані дані являли собою дискретну послідовність частотно-часового сигналу, який згідно з запропонованим алгоритмом вейвлет-аналізу був декомпозований у вигляді ієрархії декількох шарів (strata).

Для кожного із вилучених сигналів змінна часу була незалежною. Наявність різких амплітуд у кожному з поданих сигналів відповідалапевним групам аномалій.

1.2 Статистичний аналіз

Статистичний аналіз є ядром методів виявлення аномалій у мережі. До цієї групи належать такі методи:

- ланцюги Маркова;
- метод хі-квадрат (χ^2);
- метод середньоквадратичних відхилень;
- аналіз розподілів інтенсивності відправлення/прийому пакетів;
- аналіз часових рядів; пороговий аналіз.

Застосування інших поведінкових методів у контексті виявлення мережевих атак розглянуто в [5].

До методів на основі знань відносять такі методи, які в контексті заданих фактів, правил виведення та зіставлення, що відображають ознаки заданих атак, роблять дії щодо виявлення атак на основі закладеного механізму пошуку.

1.2.1 Snort

Snort є одним з найпопулярніших open-source інструментів у арсеналі системних адміністраторів під час вирішення безлічі проблем, що з забезпечення безпеки комп'ютерних хостів лише на рівні мережі. Першорядним завданням цього ПЗ є виявлення порушень у політиках мережевої безпеки, що відповідає режиму виявлення мережевих атак. Крім того, згідно з керівництвом користувача дана система здатна функціонувати як аналізатор та реєстратор пакетів. У цих режимах система відображає дані у вікно виведення (термінал) і записує дані на диск як лог-файл. Сама система розроблялася з урахуванням вимог до високої швидкодії, що в першу чергу стало можливим завдяки простоті її ядра. Ядро системи побудовано на основі порівняння за правилами та у більшості випадків використовує швидкий пошук підрядки із застосуванням алгоритму Бойєра-Мура та механізму регулярних виразів на основі бібліотеки libpcre.

До складу системи входять чотири компоненти: різні декодери пакетів, препроцесори, модуль виявлення, а також модуль логування та повідомлення про виявлені атаки. Взаємодія між компонентами здійснюється в рамках єдиного адресного простору процесу snort (з можливістю підвантаження модулів, що динамічно виконуються) і відбувається за допомогою прямих функцій викликів, які призводять до безпосереднього запуску чергового компонента. Для кожного пакета, що захоплений з мережі, викликається функція ProcessPacket, що призводить до запуску декодера (grinder) отриманого пакета. В даний момент Snort версії 2.9.8.2 підтримує такі канали, як Ethernet, loopback, raw IP, raw IPv4, raw IPv6, PPP, IEEE 802.11, IEEE 802.5 Token Ring, FDDI, Cisco HDLC, SLIP, PPP over serial with HDLC encapsulation, Linux cooked sockets, OpenBSD PF log, old OpenBSD PF log. Для кожного з цих і більш високорівневих протоколів визначено відповідні декодери пакетів, що є простими в реалізації модулями, метою яких є послідовний розбір закріплених за ними рівнів усередині структури пакета із

заповненням відповідних полів даними перехопленого пакета. Основою для реалізації цих модулів є бібліотека libpcap (winpcap), що надає весь необхідний набір функцій читання та аналізу пакетів.

Здебільшого функціонування декодерів після захоплення пакету зводиться до накладення на структуру пакета певної частини даних з автоматичним заповненням необхідних полів за допомогою виклику системної функції memcpy та інвертування байтового потоку в хостовий порядок за допомогою викликів ntohs/ntohl. У процесі декодування оброблюваних пакетів кожним модулем розставляються покажчики межі зміни сусідніх блоків всередині кожного пакета. Тому після завершення роботи одного декодера можна легко відновити роботу наступного декодера, відповідального за ініціалізацію полів відповідного протоколу у межах нової ділянки пакета. З цією метою в систему Snort жорстко запрограмована чітка послідовність процедур при обробці типових пакетів.

Для декодування вкладених в IP-пакет даних спочатку викликається функція DecodeIPOptions для аналізу IP-опцій (за умови їх наявності), за якою викликається одна з функцій DecodeTCP, DecodeUDP, DecodeICMP, DecodeIPV6, DecodeGRE, DecodeESP, DecodeAH і т.д. для відповідних значень поля protocol 6, 17, 1, 41, 47, 50, 51.

Наступний етап роботи системи Snort – запуск препроцесорів. З цією метою викликається функція Preprocess (насправді ця функція також призводить до запуску модуля виявлення атак). Препроцесори за своїм призначенням нагадують декодери, але на відміну від них здатні проводити розширений набір складніших операцій, таких як нормалізація даних, склеювання пакетів в один потік, збереження станів управління сесією. Функція RegisterPreprocessors заносить до односпрямованого списку модулі з початковою їх ініціалізацією. Так, за умовчанням системою завантажуються модулі, що виконують такі функції:

- декодування ARP-пакетів для виявлення атак ARPspoof заміною апаратних адрес (SetupARPspoof);

- нормалізація мережевих і транспортних протоколів IP, ICMP, IPv6, ICMPv6, TCP (SetupNormalizer);
- дефрагментація IP-пакетів (SetupFrag3);
- відстеження сесій та збереження їх станів для протоколів IP;
- реассемблювання вмісту TCP-потоків з урахуванням непослідовності надходження пакетів (SetupStream6);
- збирання декількох RPC фрагментованих пакетів в єдиний запис (SetupRPCDecode), виявлення атаки back orifice (SetupBO);
- аналіз HTTP-заголовків (SetupHTTPInspect);
- вимірювання теоретичної максимальної продуктивності Snort (SetupPerfMonitor);
- виявлення атак, спрямованих на сканування портів та збір інформації про вразливість сканованої системи (SetupSfPortscan).

Також серед додаткових препроцесорів можна назвати декодування команд мережевих служб SMTP, POP3, IMAP, FTP, Telnet, SSH, DNS, SSL/TLS, SIP і т.д. Функція DispatchPreprocessors відповідає за виклик кожного з таких препроцесорів зі своїм набором опцій, які вказані у конфігураційному файлі /etc/snort/snort.conf. Після роботи препроцесорів функція Preprocess викликає функцію Detect за умови, що Snort запущено як виявлення мережевих атак. До кожного з пакетів типу TCP, UDP, ICMP, IP застосовується набір правил відповідного типу у вигляді виклику функцій fpEvalHeaderTCP, fpEvalHeaderUDP, fpEvalHeaderICMP, fpEvalHeaderIP. Цим наборам правил призначається групове правило (prmFindRuleGroup) обробки пакетів найбільш загального виду відповідного протоколу. Зрештою, послідовність викликів функцій fpEvalHeaderSW призводить до обходу ієрархічного списку вкладених правил.

Модулі виведення (логування та повідомлення) призначені для приведення вихідних даних до уніфікованого вигляду. Вихідні дані можуть генеруватися, наприклад, під час обробки подій, надісланих одним із модулів виявлення під час спрацьовування одного з правил. Серед форматів

виведення, що підтримуються, можна назвати формат бази даних, XML, tcpdump, syslog.

Система Snort є одним з найкращих програмних рішень в області мережевої безпеки, що поєднує в собі легковагість, модульну розширюваність та прозору архітектуру. Крім того, в системі реалізовано безліч корисних додаткових функцій щодо аналізу мережного потоку, таких як робота в режимі офлайн з читанням збережених на диску мережевих дампів, обчислення статистичних відомостей про перехоплені пакети з угрупуванням за конкретними протоколами, підрахунок числа відфільтрованих та відкинутих пакетів, відображення витрачених у процесі аналізу трафіку тимчасових та апаратних ресурсів. Серед переваг системи також варто підкреслити наявність у відкритому доступі великої кількості готових до практичного застосування правил, що надаються.

1.2.2 Suricata

Suricata проектувалася з розрахунком на використання в обладнанні, в якому реалізовано підтримку сучасних технологій у галузі апаратного забезпечення. Розробники системи особливу увагу приділили тому факту, що система запускатиметься саме на багатопроцесорних/багатоядерних машинах. Також для досягнення цієї мети було зроблено спроби максимально оптимізувати функціонування системи, що розробляється, з орієнтацією на сучасні моделі чіпсетів CPU та GPU. Була спеціально розроблена багатопотокова архітектура, яка дозволила використовувати весь обчислювальний потенціал та міць цих машин у прийнятній мірі.

Функціонально архітектура системи включає кілька блоків: потоки, модулі та черги. Робота цих блоків поєднується конвеєрним принципом. Відповідно до цього підходу кожен із модулів здатний працювати незалежно від інших, приймаючи на кожному такті дані, відмінні від оброблюваних у цей момент іншими модулями. Здійснення такого дроблення призводить до

того, що кілька різних пакетів можуть бути оброблені системою одночасно. Зазначимо, що основні етапи функціонування Suricata при обробці пакетів такі самі, як у Snort. Відмінністю є лише те, як було зазначено вище, що етапи виконуються асинхронно. З цією метою взаємодія компонентів здійснюється через механізм буферних черг.

Серед відмінних рис системи варто виділити такі:

- масштабованість за рахунок можливості написання скриптових правил на AMBP Lua;

- прискорення обчислень за рахунок можливості часткового перенесення коду, що виконується, на графічні процесори з підтримкою технологій CUDA/OpenCL (зокрема, обчислення регулярних виразів здійснюється на GPU за умови конфігурування та наступної компіляції системи із прапором --enable-cuda);

- багатопоточність;

- відсутність прив'язки до номеру мережного порту для розпізнавання типу прикладного сервісу за рахунок наявності в системі вбудованого DPI-парсеру;

- наявність потужного аналізатора HTTP-контенту на основі бібліотеки libhtp.

Недоліками системи є досить мізерна документація та часткова підтримка правил Snort (обробляється лише обмежений набір команд). Ця система насамперед позиціонується авторами як система запобігання атакам. Акцент у цю сторону обумовлений наявністю в системі функцій блокування трафіку за допомогою бібліотек libnetfilter-queue та libnfnetlink.

1.2.3 Bro

Bro зароджувалася як дослідницький прототип у надрах міжнародного інституту комп'ютерних наук у Берклі (International Computer Science) Institute in Berkeley). Дана СВА призначена для роботи у

високонавантажених комп'ютерних мережах, орієнтована на оптимізацію обробки мережевих мереж. подій, що виникають у процесі аналізу мережевого трафіку.

Механізм, що застосовується в подібній системі, називається подієво-орієнтованим аналізом, який полягає у визначенні користувачем функцій реакції у відповідь (Callback-functions) на виникнення в системі певних мережевих подій.

Такий підхід має ряд переваг у порівнянні з традиційними списками сигнатур:

- дозволяє інтегрувати в систему власні модулі для обробки цікавих подій без необхідності їх компіляції (на відміну від модулів Snort) та перескладання всієї системи повністю;

- надає високорівневий інтерфейс для зручного моніторингу (відстеження) зміни вказаних полів мережевих пакетів та параметрів сесій;

- забезпечує простий доступ до виклику зовнішніх команд Unix-оболонки.

Серед особливостей системи варто відзначити такі:

- розширюваність. В основу роботи системи закладено можливість додавання правил користувача для аналізу мережевого потоку. Подібні правила описуються у вигляді скриптових сценаріїв внутрішньою мовою системи Bro, синтаксис і семантика якого є сумішшю мов C і Python. В рамках цієї мови в користування програмістам надається набір вбудованих у ядро мови глобальних структур даних. Змінні цих типів даних виступають у ролі фактичних параметрів при виклику функцій-обробників подій, які генеруються під час прослуховування інтерфейсу мережі. Крім того, існує можливість для конвертації правил Snort у внутрішнє представлення системи Bro за допомогою Python-скрипту snort2bro, хоча підтримка деяких останніх функціїй Snort як і раніше обмежена.

Взагалі, розширення основного функціоналу системи збудовано навколо так званих аналізаторів подій:

- ефективність. Система спочатку розроблялася як програмний продукт, який запускатиметься для аналізу мережних пакетів у високошвидкісних магістральних каналах передачі даних. З цією метою було розроблено спеціальну архітектуру системи;

- можливість аналізу протоколів різного рівня. У системі реалізована підтримка множини протоколів від канального до прикладних рівнів моделі OSI. Серед них подано основні протоколи прикладного рівня HTTP, FTP, SMTP, DNS, SSH, DHCP, транспортного рівня TCP, UDP, мережевого рівня ICMP, канального рівня ARP та ін.

Система Bro до свого складу включає наступні компоненти: підсистему перехоплення трафіку, підсистему генерації та обробки подій та інтерпретатор скриптів.

Підсистема перехоплення мережевого трафіку представлена у вигляді сніффера мережевих пакетів, що реалізований з використанням бібліотеки libpcap. На цей рівень припадає найбільший обсяг даних, тому вимоги, що пред'являються до цього компонента, полягають у необхідності швидкого аналізу мережевого трафіку і фільтрації лише свідомо необхідних пакетів. Наприклад, можна налаштувати систему для ігнорування пакетів IPv6, ARP, RARP та та прийому тільки пакетів з типом заголовка мережевого рівня IPv4 з фільтрацією по TCP-портах 21 (FTP), 22 (SSH), 143 (IMAP). BPF-фільтр у цьому випадку буде виглядати так: ether proto 0x0800 and ip proto 6 and (tcp port 21 або tcp port 22 or tcp port 143).

За допомогою застосування таких правил вдається знизити потік з'єднань лише на рівні ядра ОС тощо, а також уникнути необхідності копіювання бінарних даних у простір користувача та додаткового перемикання контексту між кільцями захисту ОС, підвищуючи загальну продуктивність системи без пропуску пакетів.

Підсистема генерації та обробки подій є ключовим компонентом системи Bro, який дозволяє виконувати зазначені в callback-функціях дії, що відповідають подій всередині аналізаторів протоколів, та контролювати

генерацію подій. Аналізатори протоколів є невід'ємними складовими даного модуля, які у процесі своєї роботи породжують різні події, що є важливими з погляду функціонування цього протоколу.

В результаті завдання вмісту callback-функцій, прив'язаних до певних подій, аналізатори протоколів здатні здійснювати цілеспрямовані дії аналізу мережевого трафіку та виявлення у ньому шкідливої активності відповідно до набору команд, зазначених програмістом. Третій компонент системи Bro – це інтерпретатор скриптів. Призначення даного модуля полягає в синтаксичному аналізі вмісту користувальницьких та системних сценаріїв, перевірці їх коректності та побудові абстрактного синтаксичного дерева. Сценарії описують різні допоміжні функції, а також обробники подій, ідентичні їм, але не мають повертаємого значення.

Кожна подія, що передається в інтерпретатор, надходить із частково скомпільованим поданням відповідного йому оброблювача. Інтерпретатор пов'язує значення фактичних параметрів всередині подій з формальними параметрами функції-обробника. Скомпільований код може виконувати такі команди, як реєстрацію повідомлень у режимі реального часу, запис даних на диск, створення нових подій, зміну внутрішнього стану для доступу та контролю до тих даних, що викликаються згодом оброблювачем подій.

Серед недоліків системи можна відзначити те, що наявність високорівневих функцій аналізу мережного трафіку може призводити до пропуску нестандартних низькорівневих видів атак (наприклад, прихованих атак, а також атак, пов'язаних із некоректно обчисленою контрольною сумою або з некоректними IP-опціями для окремих пакетів). Крім того, система не розрахована для запуску „з коробки“ і вимагає від операторів та адміністраторів попереднього грамотного налаштування та глибоких знань у галузі міжмережної взаємодії та функціонування мережних протоколів. Важливий мінус цієї СВА полягає у відсутності будь-яких графічних інструментів для перегляду записів та можливості їх збереження у БД.

1.2.4 Ossec

OSSEC є масштабованою хостовою СВА, яка виконує ряд таких функцій:

- аналіз цілісності файлів;
- перевірка реєстру Windows;
- виявлення руткітів, спроб підвищення привілеїв до рівня суперкористувача та виконання команд від його імені (тільки для UNIX);
- генерація попереджень у режимі реального часу;
- виявлення помилок сегментації (segfaults) у додатках та інших критичних подій у ядрі ОС;
- активна реакція у відповідь на події різного роду.

Основними етапами роботи системи є збір даних агентами, декодування отриманої від них інформації, аналіз та генерація попереджень відповідно до списку встановлених правил. На клієнтській стороні агенти відстежують зміни лог-файлів та відправляють їх вміст в архівованому вигляді центральному серверу з опціональним застосуванням шифрування.

Серверний компонент, у свою чергу, робить прийом, розпакування, декодування та аналіз даних та у разі спрацювання одного з правил повідомляє адміністратора системи за допомогою відправки попередження через електронну пошту або активну протидію.

При старті системи запускаються такі демони:

- ossec-authd (процес реєстрації агентів для підключення до сервера);
- ossec-analysisd (процес аналізу лог-файлів відповідно до заданих правил);
- ossec-csyslogd (процес відправлення попереджень через syslog);
- ossec-remoted (процес взаємодії сервера з клієнтськими агентами; за умовчанням для прослуховування вхідних від агентів з'єднань використовується UDP-порт 1514);

- osseclogcollector (процес моніторингу зміни контрольованих файлів та виведення повідомлень системних команд);
- ossec-agentd (процес відправлення лог-файлів на серверну сторону);
- ossec-maild (процес відправлення e-mail-повідомлень);
- ossec-execd (процес виконання активних дій: перезапуск компонентів, блокування підозрілих мережевих з'єднань);
- ossec-monitord (процес відстеження підключень до агентів та стиснення денних лог-файлів);
- ossec-syscheckd (процес перевірки зміни у файлів контрольних сум, прав доступу та власників);
- ossec-dbd (процес вставки записів із попередженнями до БД).

OSSEC має ряд недоліків, які, втім, властиві всяким хостовим СВА:

- виявляється лише постфактум проведення самої атаки;
- відсутні превентивні заходи щодо ліквідації та захисту від перших проявів мережевих атак (як наслідок першого пункту);
- відстежуються лише ті системні та мережеві служби, які ведуть запис результатів своєї роботи у вигляді лог-файлів, що обробляються цією СОА.

Серед переваг OSSEC як і будь-який хостовий СОА можна назвати:

- низький рівень хибних спрацьовувань;
- можливість виявлення атак із прихованням та зі вставкою, які не виявляються мережними СОА через специфічність реалізації мережевого стека на хості, що захищається;
- здатність аналізу контенту, що передається у зашифрованому вигляді по мережі.

Система має потужний механізм аналізу лог-файлів. Мова налаштування системи представлена у вигляді XML-розмітки і дозволяє створювати ланцюжки залежностей правил та декодерів за допомогою тегів if_sid та parent відповідно. Також на рівні декодерів за допомогою тегів regex та order можна аналізувати та отримувати окремі поля всередині лог-записів,

які можуть бути використані для формування нових тегів на рівні правил та зіставлення знайдених полів із заданим значеннями.

1.2.5 Prelude

Relude є універсальною системою, що поєднує в собі основні та необхідні функції з управління інформаційною безпекою. У їх число входять такі важливі операції:

- збір даних. Prelude відстежує дані про події безпеки як на рівні хоста, і на рівні мережі. З цією метою до системи підключаються сенсори, які виконують функції перехоплення та аналізу мережевого трафіку, моніторингу, зміни конфігураційних файлів та їх хеш-сум (контролю цілісності), перевірці вмісту лог-файлів на наявність підозрілих записів, оповіщення про вихід з ладу будь-якого пристрою або системної/мережевої служби. Логічно такі датчики видаються як СВА, модулі ФС та зовнішніх пристроїв, аналізатори журналів аудиту;
- нормалізація. Вихідні події від компонентів Prelude представляються як IDMEF повідомлення, формат яких є XML-подібний відкритий стандарт для обміну даними між пристроями системи. Дані зберігаються в єдиному централізованому сховищі, записи якого згідно з IDMEF-форматом жорстко структуровані та мають уніфіковане подання. Це дозволяє супроводжуючим розробникам позбутися написання безлічі процедур декодуючих для вихідних даних від кожного пристрою;
- агрегація. У системі здійснено можливість підключення різнорідних пристроїв від різних виробників (вендорів). Використання таких систем, як Snort, Suricata, OSSEC, Samhain, як сенсори для управління за допомогою Prelude дозволяє об'єднати результати їх роботи та уникнути аналізу специфічних лог-файлів із боку аналітиків;
- фільтрація. Подіями у системі можна порівняти певний рівень загрози (severity), який дозволяє адміністраторам звернути увагу на

найбільш небезпечні події та своєчасно вжити відповідних заходів;

- кореляція. Система отримує інформацію про погрози від кількох менеджерів управління та виявляє залежність цих даних відповідно до заданих правил кореляції;

- оповіщення про виявлені погрози. Ця функція реалізована у системі з допомогою створення звітів. Це дозволяє організаціям подавати дані про виявлені загрози у зручному візуальному вигляді.

Prelude як система, що має всі аспекти SIEM-моніторингу, належить до класу систем, відмінних від СВА.

Розробники системи не оминули питання про необхідність створення безпечного каналу для взаємодії між компонентами. Так, за наявності на хостах бібліотеки OpenSSL, що встановлюються, передача даних між хостами здійснюватиметься через шифровані з'єднання. При реєстрації нового сенсора спочатку генерується сертифікат, який буде потрібний для автентифікації джерела даних під час його наступних підключень. Далі обмін даними відбувається за допомогою тунелювання повідомлень поверх TLS/SSL. В системі є набір API для можливості швидкої реалізації сенсорів з необхідною функціональністю. За основу сенсорів береться готова бібліотека libprelude, серед функцій якої можна назвати взаємодію з Prelude-менеджером, створення шифрованого каналу, а також формування та надсилання IDMEF-повідомлень у рамках цього каналу. Для прискорення передачі повідомлень обмін даними здійснюється у бінарному вигляді відповідно до внутрішнього подання формату IDMEF, описаним у libprelude як структура на мові C. В даний момент існують байндинги до таких АМВР, як C, C++, Perl, Python, Ruby, Lua.

1.3 Порівняльний аналіз існуючих СВА

Майже всі описані СВА є активними. Винятком із цього списку є лише Bro, яка не має вбудованих засобів для запобігання атак, проте за допомогою

модуля, що йде в поставці з нею exec.bro236 можна налаштувати примусове скидання підозрілого з'єднання чи блокування трафіку лише на рівні ядра з допомогою правил iptables. Це нітрохи не збіднює цю платформу, оскільки згідно з джерелом Bro розроблялася в першу чергу саме для вивчення характеристик мережного трафіку, а не для виявлення у ньому атак.

З цією метою в ній було реалізовано потужні аналізатори протоколів, що дозволили ідентифікувати тип протоколу навіть на нестандартних портах завдяки механізму DPD (Data Packet Detection, Dynamic Protocol Detection), який аналогічний застосовуваному Suricata. Крім того, при розробці Bro далося взнаки її минуле: вона розвивалася в академічному оточенні, тому характеризується властивістю адаптивності за рахунок наявності в її основі статистичних модулів виявлення мережевих аномалій: спроб сканування портів (scan.bro), проведення DoS-атак (synflood.bro) та і т.і.

Всі представлені СВА є кросплатформовими, за винятком Bro, яка призначена для запуску тільки в Unix-подібних ОС. Також зазначимо, що для платформи Windows система OSSEC не має режиму локальної установки, тобто для цієї ОС необхідно встановлювати окремо і сервер, і агенти. Для ОС Linux, Mac OS та BSD такого обмеження немає. І в той же час саме OSSEC є єдиною з представлених СВА, здатна виявляти шкідливі процеси як наслідки успішно реалізованих атак.

За принципом побудови можна класифікувати дані системи як монолітні та компонентні. У першому випадку система представлена як єдиний бінарний файл, запуск якого засобами ОС зазвичай породжує не більше одного процесу. До них із розгляданих СВА относятся Snort, Suricata та Bro.

Компонентний підхід передбачає розбиття системи на кілька функціональних блоків, кожен з яких запускається в окремому просторі пулу адрес пам'яті, виконує конкретне завдання та спілкується з іншими на основі механізмів міжпроцесної взаємодії. До ряду таких систем належать OSSEC та Prelude. Як уже було зазначено раніше, основними елементами для створення

додаткових функцій у роботі систем Snort та Suricata є декодери та препроцесори, основа функціонування яких полягає у виклику певних функцій з бібліотек, що динамічно підвантажуються (so/dll).

Найбільш просунутим способом створення нових модулів, що розширюють функціональні можливості системи, має Bro. Вся рутинна робота з написання шаблонних файлів майбутньої програми зведено до мінімуму. Разом з вихідними кодами даної системи поставляється bash-скрипт init-plugin, призначений для автоматичної генерації початкового скелета майбутнього модуля, правил його складання та встановлення. Після компіляції модуль є готова so-бібліотека, прототипи функцій з якої загорнуті у відповідний bro-скрипт.

Наступне завантаження, ініціалізація покажчиків на функції та деалокація модулів системою зводяться до виклику функцій dlopen, dlsym та dlclose відповідно.

Серед розглянутих СВА найбільш повними характеристиками володіє Prelude. Будучи спроектованою спочатку розподіленою системою, вона підтримує гібридний моніторинг контрольованих вузлів, здійснюючи аналіз як на рівні мережі, так і на рівні хоста. Крім того, ця система є масштабованою, що дозволяє їй використовувати безліч різноманітних джерел для збору та обробки даних. Модульний принцип встановлення цієї системи також дозволяє досягти більш гнучкого налаштування кожного з її компонентів окремо.

Можливість підключення кожної з представлених СВА як сенсорів для Prelude (для Bro необхідно реалізовувати нового клієнта вручну, тому що для неї немає готового сенсора) ставить її на ранг вище решти СВА.

2 МЕТОДИ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ В СВА

2.1 Відмінності між обчислювальним та штучним інтелектами

Відповідно до тесту Тьюринга [20] (1950 р.) обчислювальна машина є інтелектуальною в термінах штучного інтелекту (ШІ), якщо вона здатна виконувати дії (роздуми, когнітивне сприйняття світу, помилкові судження тощо), подібні до такого інтелектуального суті, як людина. У рамках запропонованого Тьюрингом підходу випробувані машина і людина піддаються діалоговій екзаменації з боку людини-експерта, метою якої є виявлення сутності, що не має інтелекта. Цей тест вимагає від обчислювальної техніки не тільки здібності виконання закладених у неї алгоритмів, а й розуміння реалізованих за допомогою них дій.

Відмінності між областями ШІ та ОІ полягають у наступному:

- у той час як ОІ обробляє числове подання інформації, ІІ розглядає дані у символічному вигляді;
- ОІ аналізує структуру об'єкта у стилі „знизу вгору“, спираючись на низькорівневу інформацію про нього, а ШІ ґрунтується на побудові системи у стилі „згори вниз“, покладаючись на високоабстрактне уявлення про об'єкт;
- методи ОІ спрямовані на моделювання природних процесів або систем, пов'язаних з інтелектуальною поведінкою, а ШІ орієнтований безпосередньо на моделювання інтелектуальної, або людської, поведінки (сприйняття, міркування, навчання, спілкування, вироблення дій у складному середовищі, а також самовдосконалення) у вигляді отримання людських знань;
- якщо традиційні системи ШІ використовують явні експертні знання і точний логічний висновок, що може призводити до збільшення обсягу збережених даних і неоптимальному пошуку, то ОІ відкриває завісу над поняттям евристичних підходів, здатних здійснювати наближений

пошук оптимального рішення за менше кроків.

Розглянемо застосування теорії еволюційних обчислень та кінцевих автоматів (КА) для виявлення аномальних мережевих з'єднань. Суть використання КА полягає у побудові такої моделі, яка при спрацьовуванні зовнішнього чи внутрішнього умови здійснює перехід у одне чи кілька можливих станів. Насправді найчастіше розглядають детерміновані КА (ДКА), які характеризуються однозначністю переходу при зміні станів.

Визначимо ДКА як п'ятірку $DFA = \langle Q, \Sigma, \delta, q0, F \rangle$:

- Q – безліч станів ДКА (кінцевий); – Σ – безліч вхідних символів (вхідний алфавіт);
- $\delta : Q \times \Sigma \rightarrow Q$ – функція зміни стану;
- $q0$ – початковий стан ($q0 \in Q$);
- F – безліч кінцевих станів ($F \subset Q$).

Цей математичний апарат добре підходить для опису поведінки динамічних об'єктів. На основі наявної апріорної інформації про поточному стані об'єкта та зовнішньому впливі (умові переходу) можна спрогнозувати та змоделювати його наступний стан. Можливі два способу генерації КА з використанням еволюційних обчислень:

- генетичні алгоритми (рішення представляється у вигляді послідовності бітових символів);
- генетичне програмування (рішення представляється у вигляді дерева виразів).

Розглянемо перший підхід, який включає застосування генетичних алгоритмів для створення КА, на прикладі виявлення трьох типів атак: сканування, DoS-атаки та атаки, що характеризується великою кількістю невдалих спроб віддаленого входу в систему, що компрометується. На рисунку 2.1 показаний приклад подання простого КА, що є одним із рішень завдання про виявлення цих класів атак.

Початковим станом є стан $q0 = S0$, множина кінцевих станів складається з чотирьох елементів $F = \{S0, S5, S6, S7\}$

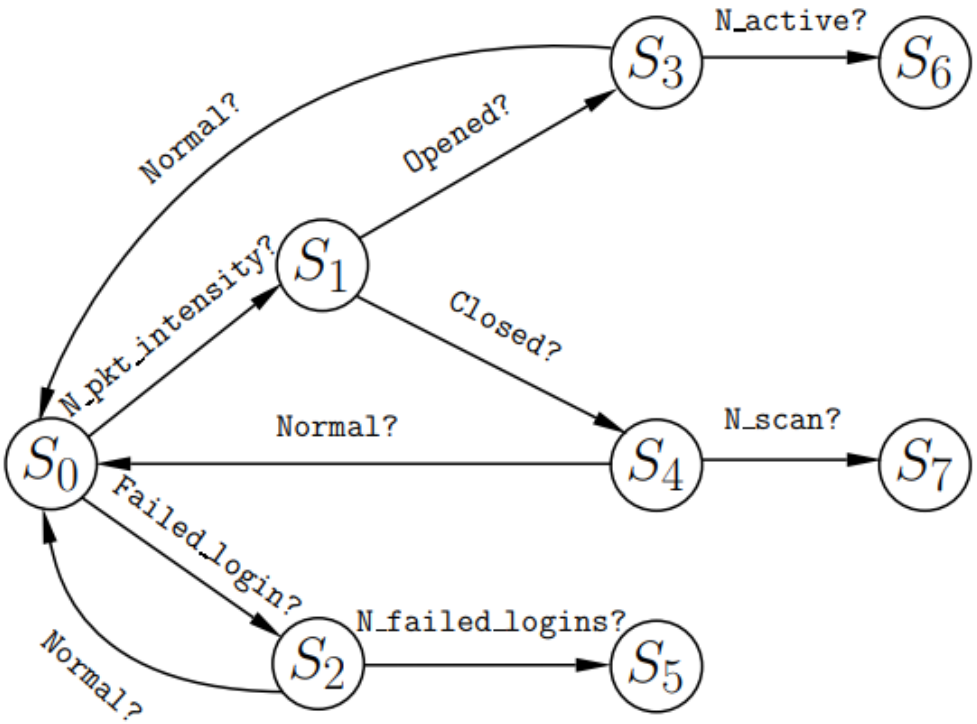

Рисунок 2.1 – Приклад КА для виявлення класу атак

Зауважимо, що у прикладі до уваги приймаються тільки TCP-пакети. Крім того, даний КА вже наведено до стандартного вигляду (здійснена нумерація вершин відповідно до обходу завширшки), який дозволяє однозначно подати його у вигляді послідовності бітових символів.

Перша послідовність із трьох бітів (000) відповідає початковому станом КА *S*0. Наступні послідовності з 6 бітів - це пари виду (перехідний стан, вхідна умова). Варто зазначити, що у згенерованих таким чином КА всі стани, крім кінцевих, повинні мати рівно два умовних переходів.

На початковій стадії генетичного алгоритму створюється популяція з кількох примірників КА. Це покоління може бути створене або довільно або на підставі будь-якого оптимізаційного критерію. В якості такого критерію може бути попередній відбір деякої кількості КА, які мають якість розпізнавання не менше зазначеної величини. Значення функції пристосованості дозволяє оцінити вклад кожного КА вирішення цієї задачі. Відбираються ті екземпляри, які мають найбільше значення функції

пристосованості. В результаті застосування оператора схрещування двох рішень створюється нове рішення, що, можливо, має більше значення функції пристосованості в порівнянні з початковими рішеннями. Для цього два предки обмінюються випадковим чином вибраними ділянками хромосом. Оператор мутації має на увазі довільну зміну деяких генів усередині хромосоми екземпляра. Нова популяція формується частково зі старого генофонду та нових представників із найбільшим значенням функції пристосованості.

Алгоритм завершується, якщо перевищено кількість ітерацій його виконання або сформована група рішень має середньоквадратичну помилку, яка не перевищує зазначеного порога.

Іншим підходом до генерації КА є генетичне програмування (рисунок 2.2).

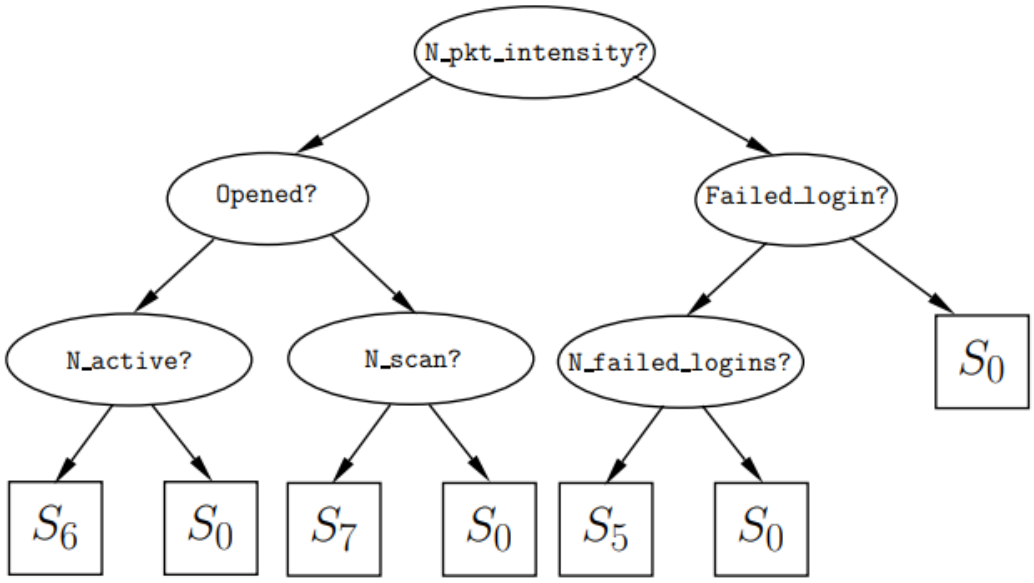

Рисунок 2.2 – Представлення КА у вигляді дерева виразів

У цьому випадку функціонування КА пропонується описувати як дерева виразів. У ролі нетермінальних вузлів у дереві виступають оператори умовних переходів. Перехід у ліву гілку піддерева виконується у разі, якщо

значення у вузлі-предку істинно, інакше виконується перехід у праве піддерево. Безліч термінальних вузлів (листя) є безліч кінцевих станів аналізованого КА, при досягненні яких аномальне з’єднання класифікується відповідно до одного з них.

2.2 Класифікація СВА

СВА – програмний або апаратний засіб, який забезпечує ряд функцій: забезпечення безпеки інформаційних ресурсів, що підлягають захисту, виявлення підозрілих та аномальних подій безпеки, виявлення фактів неавторизованого доступу до комп'ютерної системи або мережі. Основне завдання СВА полягає у забезпеченні виявлення та запобігання загрозам безпеці. СВА на кшталт реакції у відповідь поділяються на пасивні та активні. Перші спрямовані лише на виявлення факту наявності зловмисної діяльності в системі або мережі, в той час як активні СВА також здатні протидіяти та блокувати атаки.

За типом оброблюваної інформації можна виділити хостові та мережні СВА. Хостові СВА забезпечують збір інформації про різні події безпеки, що стосується даного хоста. До таких подій можна віднести виклик певних системних команд, запуск програм чи служб, зміну конфігураційних файлів. Ці СВА характеризуються здатністю виявляти атаки, спрямовані на компрометацію саме того ПЗ та обладнання, яке встановлено на хості, що захищається.

Мережні СВА обробляють інформаційні потоки, отримані від мережевих датчиків, та виявляють мережеві аномалії.

За типом архітектури СВА можна поділити на два класи: автономні та клієнт-серверні [19]. Принцип роботи автономних СВА заснований на тому, що збір інформації та аналіз трафіку та журналів реєстрації подій здійснюються на одному комп'ютерному вузлі. Тому такі системи

відрізняються швидкою швидкістю роботи і вимагають великих обчислювальних ресурсів.

Другий клас СВА, також званих розподіленими, є найбільш поширеним. Популярність використання таких систем визначається декількома факторами: модульністю, централізованим доступом та можливістю аналізу трафіку, що циркулює між усіма хостами, що спостерігаються. Правильне розміщення компонентів моніторингу трафіку у цих системах дозволяє отримувати всеосяжну картину загроз, можливих для цієї комп'ютерної мережі. Серед основних компонентів розподіленої СВА можна перерахувати наступні:

- сенсори (агенти збору первинної інформації);
- колектор; – модуль виявлення шкідливої активності;
- підсистема реагування виникнення інцидентів безпеки;
- консоль адміністрування;
- зовнішні сховища даних;
- основа правил.

На рисунку 2.3 зображено архітектуру розподіленої системи виявлення атак.

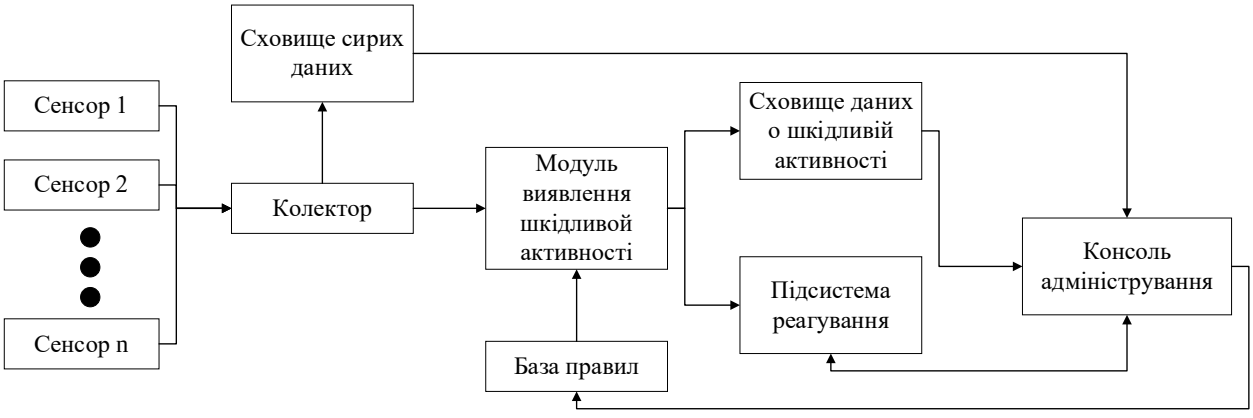

Рисунок 2.3 – Архітектуру розподіленої системи виявлення атак

Сенсори представляють собою датчики, які встановлюються в контрольовані сегменти комп'ютерної мережі чи певні хости. Колектор, в

свою чергу, приймає на вхід дані, отримані від кількох сенсорів, додатково фільтрує вхідний потік, виконує процедуру кореляції з метою подальшого виявлення розподілених атак та поміщає їх у сховище сирих даних.

Модуль виявлення шкідливої активності може бути реалізований як компонент виявлення аномалій чи зловживань. Підсистема реагування у разі виявлення підозрілих подій і мережевих з'єднань повинна здійснювати одну з таких дій:

- оповіщати адміністратора безпеки;
- скидати активне з'єднання;
- блокувати вхідні з'єднання з певною IP-адресою.

Якщо подія позначається як підозріла, її атрибути містяться в сховище даних про шкідливу активність для можливості подальшого аналізу з боку адміністратора.

Консоль адміністрування є окремо встановленим комп'ютером або віртуальним хостом, який містить функції віддаленого конфігурування та доступу до окремих компонентів системи. Зокрема, має бути надана можливість перегляду інцидентів безпеки, для яких було згенеровано сигнал тривоги, та сирих даних для детальнішого вивчення причин спрацьовування модуля виявлення атак.

2.3 Вимоги до СВА

Сформулюємо наступні вимоги:

- можливість аналізу заголовків мережевих пакетів;
- можливість аналізу вмісту окремих мережевих пакетів, дефрагментованих послідовностей IP-пакетів та TCP-потоків;
- можливість збирання фрагментованого трафіку;
- можливість виявлення прихованих атак, атак зі вставкою;
- можливість обробки пакетів, що порушують стандартну поведінку сесії TCP.

2.4 Імунні системи

Імунна система являє собою сукупність клітин та молекул, скоординована діяльність яких полягає у здійсненні основних функцій - підтримки стабільної роботи (гомеостазу) та захисту структурної цілісності організму від багатьох патогенних мікроорганізмів (вірусів, бактерій, паразитів). При попаданні в нього чужорідного тіла відбувається активація захисних білкових молекул, і посилюється відповідна активна реакція, яка полягає у виробленні особливих клітин, здатних напряму протидіяти чи здійснювати непряму допомогу у нейтралізації цього тіла. У той же час сама імунна система має складні саморегулюючі процеси, які полягають в інгібуванні активності породжених нею клітин.

Для імітації поведінки компонентів та процесів їх взаємодії в природної імунної системи було запропоновано безліч математичних концепцій та моделей, об'єднаних загальною спрямованістю у вигляді рішення задач інтелектуальної обробки даних напрямку обчислювального інтелекту, в якому досліджуються натхненні ідеями про імунні положення підходи, отримав назву штучних, або обчислювальних, імунних систем, які на даний момент мають широку популярність у багатьох прикладних галузях інформатики.

Розглянемо деякі з існуючих дослідницьких прототипів мережевих СВА, побудованих на основі принципів штучних імунних систем.

Зазначимо, що всі з цих систем є розподіленими як найбільш універсальний та поширений підхід, що використовується для побудови СВА та організації її складових компонентів у єдину керуючу систему.

Система libtissue надає клієнт-серверне оточення для розробки та тестування імунних алгоритмів, що базуються на принципах теорії небезпеки. Роль клієнтів полягає у збиранні сигналів та антигенів, а також їх передачі на серверну сторону за протоколом SCTP. Сервер, у свою чергу, розміщує популяції клітин разом із образами об'єктів, отриманих від клієнтів,

усередині спеціально виділеного блоку – відділення тканини, в якому клітини, сигнали та антигени взаємодіють між собою за заданим користувачем алгоритмом. Зпропонована система використовується для виявлення SYN-сканувань, що виконуються без повного встановлення з'єднання. З цією метою було виділено сім сигналів, чисельно виражених різними значеннями середнього розміру, швидкості та інтенсивності передачі мережевих пакетів спеціальних типів, а також ознакою множинних віддалених входів в контрольовану систему під обліковим записом суперкористувача.

Система LISYS (Lightweight Intrusion detection SYStem) представляє собою СВА, яка орієнтована працювати виключно у широкомовних комп'ютерних мережах. Для виконання завдань розподіленого моніторингу мережевого оточення вона використовує як основу імунні детектори, представлені у вигляді двійкових рядків, та правила r-безперервних бітів.

Відповідно до цього правила, якщо структури навченого детектора та тестового об'єкта збігаються щонайменше в позиціях та послідовно йдуть один за одним, то вважається, що об'єкт являє собою аномалію (антиген). Кожен детектор кодується як кортеж, що містить IP-адреси джерела і призначення, і навіть службу TCP-з’єднання. Набір даних для навчання представлений аналогічним способом, де кожен елемент відповідає з'єднанню, чиї представлені характеристики спостерігаються з високою частотою локальної мережі. І навпаки, якщо трійка задає рідкісне за появою з'єднання, вона належить до набору „чужих“ об'єктів.

Архітектура мережної системи виявлення атак, представленої раніше, є дворівневою і включає до свого складу первинну СВА та кілька вторинних СВА. Кожна з цих СВА забезпечується комунікатором для можливості обміну даними по мережі один з одним. Первинна СВА сприймається як кістковий мозок чи тимус, що генерує зрілі імунні детектори. Для створення таких детекторів усередині цієї СВА закладено процеси еволюції. бібліотеки генів та негативного відбору. Роль першого процесу полягає у створенні та

поповненні спеціального сховища атрибутів детекторів, що генерують події про виявлення аномалій мережі, – бібліотеки генів. Додавання нового запису до бібліотеки генів при досягненні її максимальних розмірів залежить від ступеня придатності відповідного детектора, що дозволяє штучній імунній системі динамічно навчатися на актуальних даних, зібраних у різні моменти часу.

Роль другого процесу полягає у формуванні набору детекторів, попередньо згенерованих на основі даних бібліотеки генів та операторів мутації та згодом відфільтрованих за алгоритмом негативного відбору, та передачі його у вторинні СВА. У вторинних СВА виконується процес клональної селекції тих детекторів, які виявили аномальну активність у мережевому трафіку. Одна з копій клонованих детекторів зберігається як детектор пам'яті на поточній СВА, а інші поширюються інші СВА.

2.5 Розробка моделі ШІС

Розроблена модель штучної імунної системи на базі еволюційного підходу $AISEA$ Approach представляється так:

$$AISEA = \langle \mathcal{DT}, \mathcal{DM}, \mathcal{SA}, \mathcal{SN}, \mathcal{G}, R, \Psi \rangle, \quad (3.1)$$

де $\mathcal{DT} \subset \mathcal{D}$ - набір тимчасових імунних детекторів, $\mathcal{DM} \subset \mathcal{D}$ - набір імунних детекторів пам'яті, $\mathcal{SA} \subset \mathcal{S}$ — навчальний набір, що складається з аномальних екземплярів („чужих“ об'єктів), $\mathcal{SN} \subset \mathcal{S}$ — безліч, що складається з нормальних екземплярів ("своїх" об'єктів), $\mathcal{D} = \mathcal{DT} \cup \mathcal{DM}$ — набір імунних детекторів, $\mathcal{G} = \{G1, \ldots, GK\}$ - стратегії генетичної оптимізації імунних детекторів.

Кожен імунний детектор $d \in \mathcal{D}$ представляється як кортеж наступного виду:

$$d = \langle representation, threshold, life\_time, state \rangle, \qquad (3.2)$$

На початку свого розвитку кожен детектор ініціалізується довільним чином відповідно до свого внутрішнього представлення. В процесі функціонування СВА тимчасові детектори здійснюють запис параметрів атак в оновлюваний набір аномальних даних $\mathcal{SA}^*$ у разі їх виявлення. Усі імунні детектори, крім детекторів пам'яті, наділені кінцевим терміном життя. В якості внутрішнього представлення імунних детекторів були обрані мережі Кохонена ($representation = NeuralNetp$). Такі мережі є дворівневими структурами, в яких вхідні сигнали, розподілені на зовнішній шар мережі, що порівнюються з вагами нейронів вихідного шару.

Загальний підготовчий процес побудови штучних параметрів імунної системи $AISEA$ може бути описаний наступним чином:

- вибір внутрішньої структури для кожного детектору $d \in \mathcal{D}$: $representation$;

- формування навчального набору даних, що містить заздалегідь $\mathcal{SA}$ відібрані „чужі“ об'єкти;

- формування тестового набору даних, що містить заздалегідь відібрані „свої“ об'єкти;

- вибір стратегії генетичної оптимізації імунних сенсорів;

- вибір алгоритму навчання R імунних детекторів $\mathcal{D}$ залежно від їхнього внутрішнього уявлення;

- вибір правила відповідності Ψ між імунним детектором та вхідним об'єктом.

Розроблена модель штучної імунної системи – це набір імунних детекторів, представлених у вигляді тимчасових детекторів та детекторів пам'яті, в сукупності із заданим алгоритмом їх навчання, який приймає як вхідні аргументи детектор d, що настроюється, підмножини двох наборів даних (набору $\mathcal{SA}$, що складається з „чужих“ об'єктів, та набору $\mathcal{SN}$, що складається з "своїх" об'єктів), а також стратегію генетичної оптимізації.

Причому набір даних $\mathcal{SA}$ призначається для першої попередньої налаштування імунних детекторів, а роль набору даних $\mathcal{SN}$ полягає у фільтрації навчених детекторів.

Стратегії генетичної оптимізації імунних детекторів включають деякий набір генетичних операторів (кросовера, мутації, інверсії) та їх комбінацій для зміни параметрів імунного детектора після його клонування. Правило навчання імунних детекторів є двокроковою процедурою. На першій фазі імунні детектори навчаються виключно на елементах набору даних $\mathcal{SA}$ і піддаються клонованій селекції, в процесі якої створені копії імунних детекторів мутують згідно з обраною стратегією $G' \in \mathcal{G}$.

З набору детекторів, представленого виробленим потомством та вихідним детектором, відбираються тільки ті детектори, які є найбільш придатними щодо до елементів набору $\mathcal{SA}$ згідно з вибраним правилом відповідності Ψ. Дана фаза повторюється кілька разів на формування напівзрілих детекторів. на другій фазі навчання ці детектори перевіряються на відповідність „своїм“ об'єктам: ті з них, які помилково активуються, знищуються, заново ініціалізуються та навчаються. В рамках цієї моделі кожен імунний детектор піддається кільком етапам диференціювання.

2.6 Алгоритм генетико-конкурентного навчання мережі Кохонена

Карта Кохонена (рисунки 2.4, 2.5) – це нейронна мережа без зворотних зв'язків, в якій використовується алгоритм навчання без вчителя. За допомогою процесу, іменованого самоорганізацією, карта Кохонена утворює топологічне представлення вхідних даних, що аналізуються з нейронів, одержуваних на виході. Карту Кохонена можна навчити дізнаватися або знаходити взаємозв'язки між входами і виходами або організовувати дані таким чином, щоб виявляти в них раніше невідомі образи або структури.

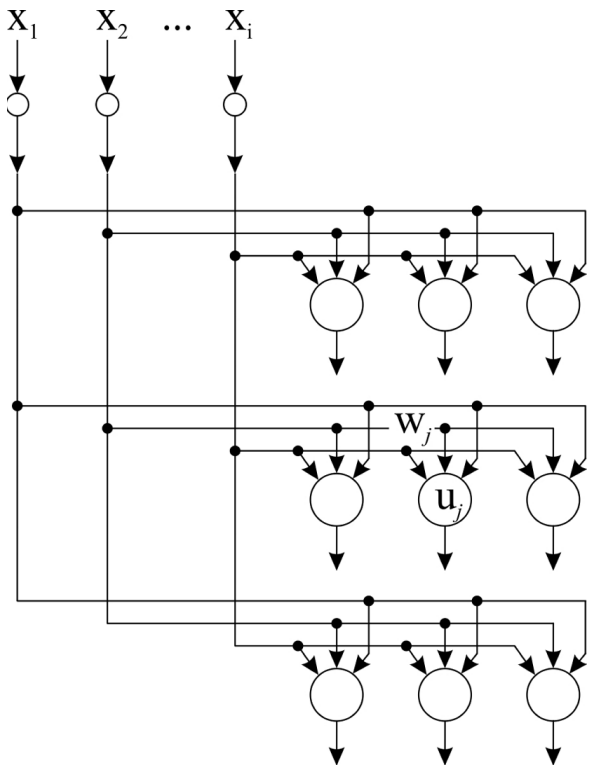

Рисунок 2.4 – Карта Кохонена

Для інтеграції мереж Кохонена в імунну модель $AISEA$ був розроблений модифікований алгоритм конкурентного навчання мережі Кохонена, доповнений введенням генетичних операторів для схрещування та мутації вагових коефіцієнтів окремих нейронів на вихідному шарі мережі.

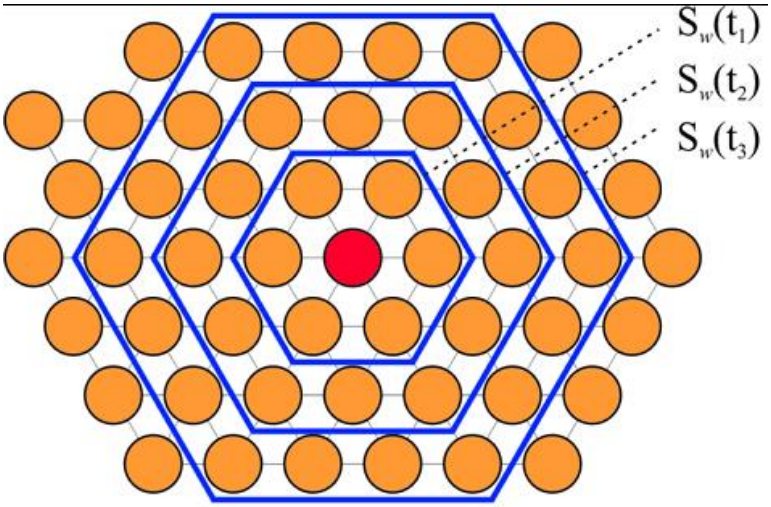

Рисунок 2.5 – Карта Кохонена

Алгоритм.

Завдання параметрів мережі Кохонена (розміру вихідної решітки $I \times J$, числа епох навчання $T > 1$, початкової ширини околиці нейронів σ_0, коефіцієнтів τ, κ_0, η), занулення лічильника поточних ітерацій := 0, ініціалізація вагових коефіцієнтів w_{ij} нейронів вихідної решітки випадковим чином, підготовка навчальних даних $\{x_k\}M$ k=1, вибір стратегії оптимізації G' вагових коефіцієнтів нейронів вихідної ґрати.

Обчислення поточної ширини околиці σ.

Ініціалізація поточного набору активних нейронів V.

Виконання кроків 1–8 для кожного вектора x_k (k = 1, . . . ,M).

Нормалізація вагових коефіцієнтів нейронів w_{ij} за допомогою покомпонентного поділу на $\|w_{ij}\|$

Нормалізація вектора x_k за допомогою покомпонентного поділу на $\|x_k\|$.

Обчислення відстаней між вектором xk та кожним ваговим вектором wij нейрону:

$$d_{ijk} = D(\boldsymbol{x_k}, \boldsymbol{w_{ij}}) = \sqrt{\sum_{l=1}^{n} (x_{kl} - w_{ijl})^2}$$

Після модифікації вагових коефіцієнтів при пред'явленні навчального вектора x_k або після виконання кількох епох конкурентного навчання додатково застосовується на основі генетичного алгоритма стохастична оптимізація вагових коефіцієнтів визначених нейронів вихідний грати карти Кохонена.

З цією метою ваги кожного нейрона, що знаходиться на околиці поточного нейрона-переможця і жодного разу не активувався, видаються як послідовність генів, що виступають у ролі мінімальної одиниці для вхідного аргументу оператора схрещування. В результаті виконання цього оператора формується пара нових хромосом, у яких переставлені місцями довільно вибрані ділянки ген батьківських хромосом. Кожен ген є набором бітів, який

можна розглядати як окремий компонент вектора, асоційованого з відповідним нейроном-переможцем або одним з нейронів, що лежать у його околиці. При використанні оператора мутації здійснюється перестановка пари випадково обраних бітів всередині одного гена, при використання оператора інверсії відбувається інвертування значення випадково вибраного біта. Обидва ці оператори застосовуються тільки до частини мантиси 64-бітного "речового гена". Для імітації процесу еволюції нейронів було розроблено кілька способів генерації поколінь, що породжуються ними.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СВА

3.1 Сигнатурний аналіз в СВА

Сигнатурний аналіз як процес виявлення мережевих атак включає себе широкий клас правил, спрямованих на порівняння значень певних полів (або обчислених мета-атрибутів) пакетів з рядом сигнатур, заданих користувачем чи закладених статично у систему. Самі сигнатури можуть представлятися як (I) ключові слова (шаблонні підрядки) або (II) команди (функції з аргументами), які вказують на аномальну дію в мережі.

У разі I виконується послідовний перегляд ключових слів, зазначених у сигнатурному правилі, і для кожного з ключових слів перевіряється його посимвольне або побайтове входження в аналізований рядок, що міститься у захопленому мережевому пакеті. У разі II сигнатури позначаються неявно через результат виклику деякою системною або користувальницькою функції, яка повертає значення мета-атрибуту, що підлягає подальшому порівнянню.

У той час як сигнатури типу I засновані на простій ідеї пошуку шаблонної підрядки в кожному пакеті і вимагають від розробника вибору алгоритму, що задовольняє найменші витрати часу, сигнатури типу II зазвичай використовуються з метою акумулювання статистичних показників, обчислених для декількох певних пакетів або сполук.

3.2 Архітектура розробленої СВА

Розроблена СВА є розподіленою і складається з кількох клієнт-сенсорів та одного сервер-колектора. На кожен контрольований вузол у мережі встановлюється два програмні компоненти: балансувальник трафіку та мережевий аналізатор. Роль першого полягає у розподілі навантаження між

декількома внутрішніми фіктивними інтерфейсами ОС. Функції аналізатора - дефрагментація IP-пакетів, формування TCP-сесій (TCP-реассемблювання), виділення атрибутів мережевого трафіку, включаючи збір статистичних показників за вхідними IP-адресами, вимірювання інтенсивності прийому спеціальних пакетів. Крім того, до завдань цього компонента додано виявлення явних порушень на рівні мережі.

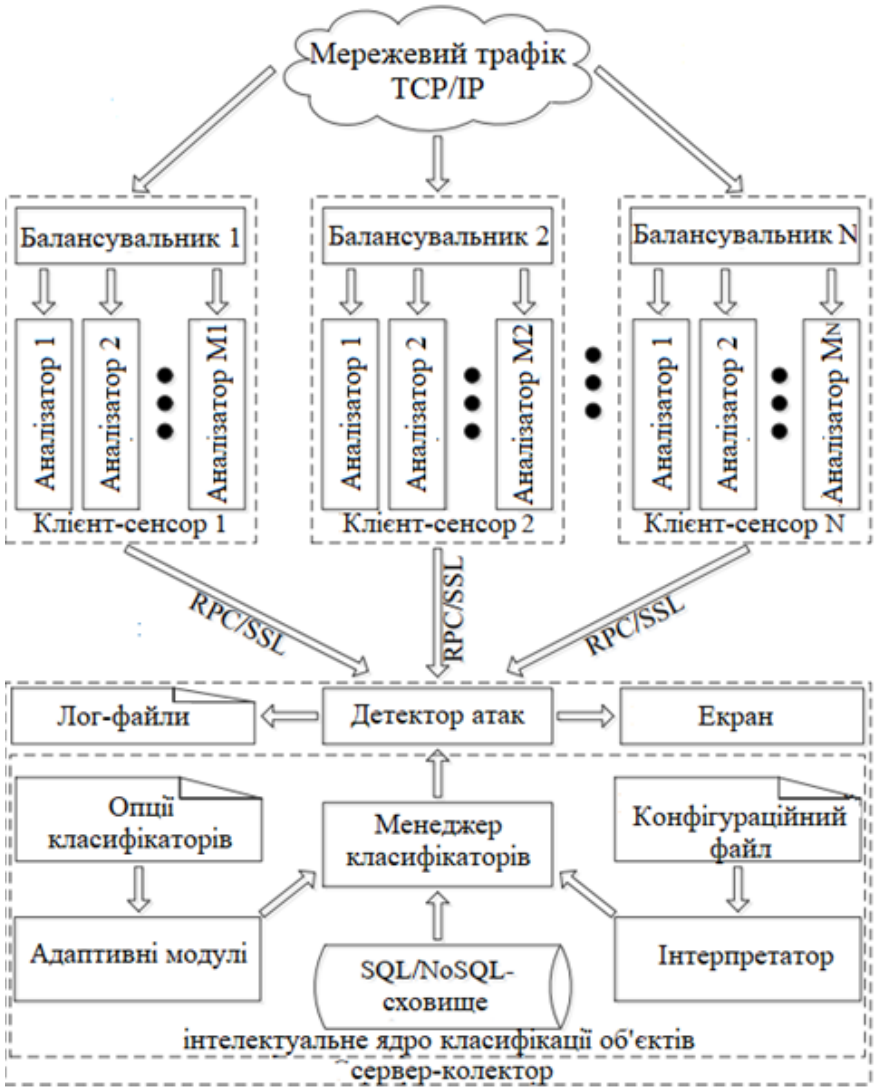

Рисунок 3.1 – Архітектура розробленої СВА

Залежно від мережевого навантаження на сенсорі може бути запущено кілька аналізаторів, кожен з яких прослуховує мережевий трафік на певному інтерфейс, що є вихідним для балансувальника. Серверна частина виконує

функції прийому даних від клієнтських сенсорів, взаємодія з якими здійснюється за допомогою механізму RPC із шифруванням SSL; для цього використовувалися компілятор rpcgen та deb-пакет libgnutls-openssl-dev.

Адаптивні модулі, представлені у вигляді динамічних плагінів та побудовані на основі методів OI, реалізують навчання відповідних класифікаційних моделей і завантажуються перед запуском детектора атак за допомогою допоміжних компонентів (інтерпретатора та менеджера класифікаторів), що входять у складі інтелектуального ядра класифікації об'єктів. У разі виявлення атаки опціонально здійснюється віддалене налаштування правил фаєрволу iptables на клієнтських сенсорах за допомогою команд, що передаються RPC/SSL.

Невід'ємною частиною у функціонуванні будь-якої мережної СВА є мережний стек (підсистема обробки мережних пакетів) з реалізацією IP-дефрагментації та TCP-реассемблювання. Аналізатор отримав приставку „подійно-орієнтований“ завдяки вбудованому в його функціональність виклику функцій користувача при виникненні певних подій, пов'язаних із захопленими з мережі пакетів.

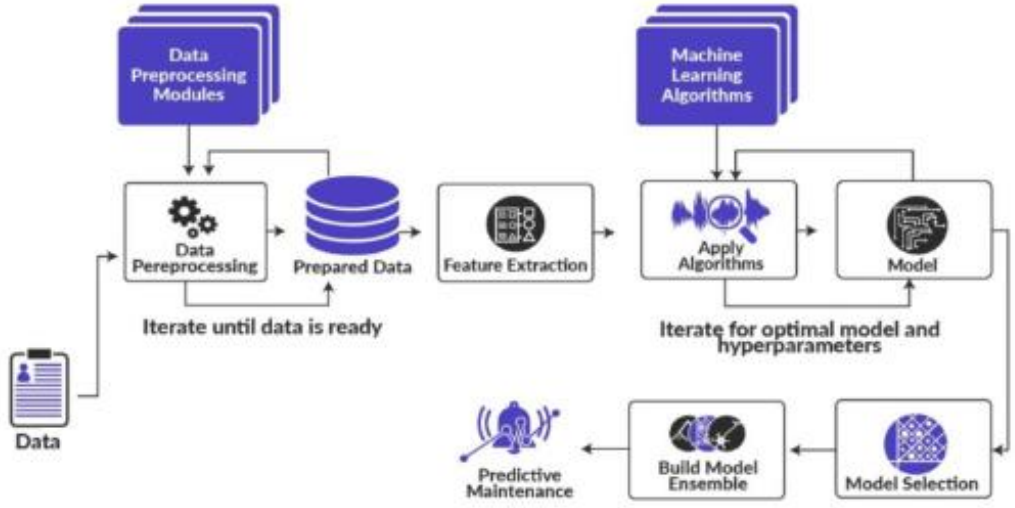

Рисунок 3.2 – Узагальнена архітектура ПЗ блоку розробленої моделі

Усього таких подій було виділено 32, опис кожного з яких представлено в таблиці 6 у порядку зменшення черговості виклику (пріоритету) відповідних їм callback-функцій. Перший компонент, представлений у двох виконаннях, – динамічна бібліотека libnetpcap.so (статична бібліотека libnetpcap.a), побудована на основі бібліотеки libpcap.so і надає весь необхідний API з підтримкою обробки мережевих подій та можливістю реєстрації відповідних їм callback-функцій.

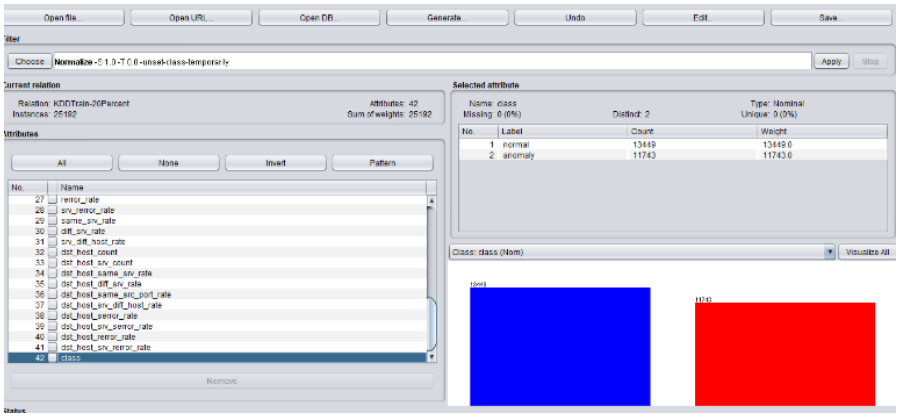

Рисунок 3.3 – Результати роботи

Рисунок 3.4 – Результати роботи

Другий компонент – це бінарний ELF-файл, що виконується netpcap_sensor, отриманий за допомогою динамічної лінківки з бібліотекою libnetpcap.so (статичної лінківки з бібліотекою libnetpcap.a). Саме другий

компонент займається вилученням 106 параметрів, що характеризують мережеві з'єднання; обчислення цих параметрів виконується у представлених callback-функціях. Зручність такого модульного рішення полягає в тому, що коли виникає необхідність додавання або зміни будь-яких мережевих параметрів, ядро аналізатора, представлене бібліотекою libnetpcap.so (libnetpcap.a) залишається статичним, а повторної компіляції піддається тільки вихідний файл netpcap_sensor.

ВИСНОВКИ

Проведено аналіз методів виявлення мережних аномалій з використанням штучних нейронних мереж та штучних імунних мереж. Розроблено модель штучної імунної системи на основі еволюційного підходу. Модель характеризується наявністю дворівневої процедури навчання та тестування, а також дозволяє враховувати динамічну природу мережного трафіку за допомогою перенавчання імунних детекторів як з часом, так і у відповідь на виявлені в мережному трафіку аномалії. В якості детекторів запропоновано використання класичного апарату штучних нейронних мереж типу карт Кохонена. Розроблено архітектуру розподіленої СВА, побудованої на основі гібридизації методів ОІ та сигнатурного аналізу. СВА характеризується можливістю гарячої вставки коду, що виконується за рахунок завантаження плагінів, представлених у вигляді бінарних бібліотек і вбудованих динамічно в ядро СВА.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hande Alemdar, Vincent Leroy, Adrien Prost-Boucle, and Fr´ed´eric P´etrot. Ternary neural networks for resource-efficient AI applications. In International Joint Conference on Neural Networks, 2017.

2. Genevera Allen. Sparse higher-order principal components analysis. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2012.

3. Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Fixed point optimization of depth convolutional neural networks for object recognition. In Acoustics, Speech, and Signal Processing (ICASSP), International Conference on, 2015.

4. M. Astrid and Seung-Ik Lee. Cp-decomposition with tensor power method for convolutional neural networks compression. In Big Data and Smart Computing, 2017.

5. Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

6. Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In International Conference on Computer Vision (ICCV), 2015.

7. Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. LCNN: Lookup-based convolutional neural network. Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

8. Блохін О.О., Міхаль О.П., Щепка О.О. Методи виявлення мережних аномалій з використанням штучних нейронних мереж// Проблеми інформатизації : десята міжнародна науково-технічна конференція. Черкаси – Баку – Бельсько-Бяла – Харків, 2022, т.2, с.119.

9. Платонов, В. В. Програмно-апаратні засоби захисту інформації. - М.: Видавничий центр «Академія», 2013. - 336 с.

10. Боршевніков А . Е. Мережеві атаки. Види. Способи боротьби. – Уфа. 2011.- С. 8-13.

11. Гамаюнов Д.Ю. Виявлення комп'ютерних атак на основі аналізу поведінки об'єктів: автореф. дис. ... канд. фіз.-мат. наук: 13.05.11. – М., 2007. – 89 с

12. А. А. Браницький, І. В. Котенко, Аналіз і класифікація методів виявлення мережевих атак, Тр. СПІІРАН, 2016, випуск 45, 207-244

13. Хайкін Р . Нейронні мережі: повний курс, 2-е видання. пер. з англ. / Р.Хайкін. - М .: Видавничий дім «Вільямс», 2006. - 1104 с.

14. Горбаченко В.І. мережі і карти Кохонена. URL: http: //gorbachenko.self- organization.ru/articles/Self-organizing_map.pdf

15. Самоорганізуючі карти. пер. з англ. / Т. Кохонен - М .: БИНОМ. Лабораторія знань 2012. - 655 с .

16. Венкатачалам. V. Порівняння продуктивності системи виявлення вторгнень класифікатори з використанням різних методів редагування ознак // SELVAN S Erode Sengunthar Машинобудівний технікум. 31 березня 2015. – с. 19

17. Мадбулі А. Модель вибору релевантних функцій з використанням аналізу даних для Система виявлення вторгнень / Madbouly, A. // Gody, A. // Barakat, T International Журнал інженерних тенденцій і технологій (IJETT) - Том 9, номер 10 - березень 2014. – С. 12

18. Керівний документ. Методологія функціонального моделювання IDEF0. Розроблено Науково-дослідним Центром CALS - технологій «Прикладна Логістика». Держстандарт. - М .: ІПК Видавництво стандартів, 2000. - 75 с.

19. Вендров А.М. CASE-технології. Сучасні методи і засоби проектування інформаційних систем.. - М .: Фінанси і статистика, 1998. – 98 с.

20. Забезпечення комплексної безпеки підприємств: проблеми та рішення: с. збірник тез доповідей IV Міжнародної науково-практичної конференції. Рязан. держ. радіотехн. ун-т; Рязань, 2016 - 144 с.

21. Басалаєва, Ю.С. Вибір інструментів Data Mining для аналізу результатів дистанційної освіти / Ю.С. Басалаєва // Сучасні матеріали, техніка та технологія. – 2015. – № 2. – С. 22 - 25.

22. В.Й. Нірмал і Д.І.Г. Amalarethinam, «Паралельна реалізація Big Алгоритми попередньої обробки даних для аналізу настроїв даних соціальних мереж», International Journal of Fuzzy Mathematical Archive, Vol. 6, № 2, стор.149-159, 2015.

23. K. M. Ting, T. Washio, J. R. Wells, F. T. Liu, S. Aryal, “DEMass: a new оцінювач щільності для великих даних», Knowledge & Information Systems, Vol. 35, стор. 493– 524, 2013 рік