МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

КУЗНЄЦОВ ІВАН ОЛЕГОВИЧ

Допускається до захисту:
в. о. завідувача кафедри
прикладної математики та
кібербезпеки,
д – р філософії з математики,
_____А.В. Луценко
«__»_____20__ р.

ВИКОРИСТАННЯ МЕТОДІВ ГЛИБОКОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АРТ-АТАК

Спеціальність 125 Кібербезпека

Кваліфікаційна (бакалаврська) робота

Керівник:
канд. техн. наук,
доцент, доцент кафедри прикладної
математики та кібербезпеки,
Загоруйко Л. В.

Оцінка:___/___/_____

(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК:_____

(підпис)

Вінниця - 2024

АНОТАЦІЯ

Кузнєцов І. О. Використання методів глибокого навчання для виявлення APT-атак. Спеціальність 125 «Кібербезпека». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (бакалаврській) роботі досліджено використання методів глибокого навчання для виявлення APT-атак. Проаналізовано моделі APT-атак, зокрема модель Cyber Kill Chain, та сучасні підходи до їх виявлення. Запропоновано метод виявлення APT-атак на рівні операційних систем з використанням методів кластеризації. Розроблено експериментальний програмний комплекс для емуляції поведінки системних користувачів та застосування методів глибокого навчання, зокрема рекурентних нейронних мереж, для ефективного виявлення шкідливої активності. Проведені експерименти продемонстрували високу ефективність запропонованих рішень, з точністю виявлення шкідливої активності до 92%.

Ключові слова: APT-атаки, глибоке навчання, кібербезпека, системна активність, машинне навчання, рекурентні нейронні мережі.

67 с., 4 табл., 19 рис., 26 джерел.

ANNOTATION

Kuznietsov I. O. Using deep learning methods to detect APT attacks. Speciality 125 ‘Cybersecurity’. Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

In the qualification (bachelor's) work, the use of deep learning methods for detecting APT attacks is investigated. APT models, in particular the Cyber Kill Chain model, and modern approaches to detecting them are analysed. A method for detecting APT attacks at the operating system level using clustering methods is proposed. An experimental software package has been developed to emulate the behaviour of system users and apply deep learning methods, in particular recurrent neural networks, to effectively detect malicious activity. The experiments have demonstrated the high

efficiency of the proposed solutions, with an accuracy of detecting malicious activity of up to 92%.

Keywords: APT attacks, deep learning, cybersecurity, system activity, machine learning, recurrent neural networks.

67 p., 4 tables, 19 figures, 26 sources.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

APT - Advanced Persistent Threat (Стійка цілеспрямована загроза)

IDS - Intrusion Detection System (Система виявлення вторгнень)

ML - Machine Learning (Машинне навчання)

DL - Deep Learning (Глибоке навчання)

RNN - Recurrent Neural Network (Рекурентна нейронна мережа)

OS - Operating System (Операційна система)

DBSCAN - Density-Based Spatial Clustering of Applications with Noise (Алгоритм кластеризації DBSCAN)

PID – Process IDentificator (ідентифікатор процесу)

PPID – Parent PID

CPID – Child PID

ВСТУП

З розвитком інформаційно-комунікаційних технологій з'являється все більше способів проведення успішних атак на віддалені інформаційно-технологічні інфраструктури.

Проблема виявлення шкідливої активності (malicious activity) стає сьогодні все більш актуальною. Однак зростаюче розмаїття серверів, протоколів, форматів і обсягів даних робить дедалі складнішим виявлення атак до початку шкідливої діяльності.

Стандартним рішенням є впровадження алгоритмів машинного навчання (machine learning) в системи виявлення вторгнень (IDS - Intrusion Detection System). Цей підхід використовується для виявлення відомих атак, таких як: зонди (probes), R2L, DdoS тощо.

Впровадження алгоритмів машинного навчання у великих інформаційних інфраструктурах з високою щільністю нестандартизованого трафіку знижує швидкість і складність аналізу. Саме це робить APT (Advanced Persistent Threat) атаки успішними. В інформаційному середовищі складно (а іноді й неможливо) відрізнити сервісну та користувацьку активність від нестандартної, шкідливої.

Актуальність дослідження зумовлена тим, що широкий спектр алгоритмів машинного навчання та методів аналізу даних надає можливості для вдосконалення методів виявлення APT-атак. На сьогоднішній день APT-атаки є однією з найбільш небезпечних загроз для великомасштабних IT-інфраструктур.

Різноманіття підходів машинного навчання, таких як глибокі нейронні мережі, методи кластеризації, аналізу аномалій тощо, дозволяє розробляти більш ефективні системи детектування складних, цілеспрямованих кібератак типу APT. На відміну від традиційних сигнатурних методів виявлення, методи глибокого навчання здатні виявляти приховані, еволюційні моделі атак, що не описуються чіткими правилами.

Метою дослідження є розробка ефективних методів використання глибокого навчання для виявлення APT-атак, а також удосконалення їх застосування в системах виявлення вторгнень.

Для досягнення даної мети необхідно виконати такі **завдання**:

1. Провести ґрунтовний аналіз існуючих методів та підходів до виявлення APT-атак, визначити їх переваги, недоліки та обмеження.
2. Розробити та обґрунтувати нові методи виявлення APT-атак на основі сучасних алгоритмів глибокого навчання, спрямовані на підвищення ефективності кібербезпеки.
3. Дослідити можливості інтеграції розроблених методів глибокого навчання в системи виявлення вторгнень (IDS) для підвищення їх здатності протидіяти складним, цілеспрямованим кібератакам.

Об’єктом дослідження є системна діяльність, процеси обміну даними в інформаційному середовищі та механіки реалізації APT-атак, що дозволить розробити ефективні методи їх виявлення на основі сучасних підходів машинного навчання.

Предметом дослідження є методи виявлення APT-атак в операційних системах, алгоритми машинного навчання та їхні якісні показники.

У ході дослідження було застосовано наступні **методи:**

1. Ґрунтовний аналіз наукової літератури за тематикою виявлення APT-атак для вивчення існуючих підходів та їх обмежень.
2. Класифікація APT-атак за різними критеріями, зокрема за характером, технологіями реалізації, цілями тощо.
3. Статистичний аналіз показників ефективності існуючих методів виявлення атак даного типу.
4. Експериментальні дослідження з використанням бібліотек для збору, обробки та аналізу інформації, зокрема наборів даних (семплів), які містять відомості про користувацьку та зловмисну активність.

Практичне значення полягає у дослідженні нових методів виявлення APT-атак на основі сучасних підходів машинного навчання, зокрема глибокого

навчання, це дозволить значно підвищити ефективність систем кібербезпеки у протидії складним, цілеспрямованим кібернетичним загрозам.

РОЗДІЛ 1. APT-АТАКИ, МЕТОДИ ЇХ ВИЯВЛЕННЯ

1.1. Моделі APT-атак

Сучасні кіберзагрози характеризуються високим рівнем складності та цілеспрямованості. Одним із таких видів загроз є Advanced Persistent Threats (APT) - складні, тривалі та приховані атаки, спрямовані на отримання несанкціонованого доступу до важливих інформаційних активів. APT-атаки відрізняються від традиційних кібератак своєю складністю, багатоступеневістю та наполегливістю зловмисників у досягненні кінцевої мети.

APT-атака — термін широко використовується як кібер-загроза (кібер-атака) для комплексної, цілеспрямованої і ефективної атаки на критичні IT інфраструктури. Одну з перших моделей APT-атак, що набула поширення у 2011 році, було запропоновано компанією Lockheed Martin [11, с. 2]. Lockheed Martin ввела новий термін cyber kill chain, який є частиною моделі Intelligence Driven Defence [5] для виявлення та запобігання процесів кібервторгнення.

Ця модель визначає, що повинен зробити зловмисник для досягнення своїх цілей, атакуючи мережу, витягуючи дані та підтримуючи свою присутність у мережі. Завдяки цій моделі блокування зловмисників на будь-якому етапі, як відомо, порушує весь ланцюжок атак.

Згідно з наведеною нижче моделлю, кінцева точка - це неминуча точка, через яку проходять усі атаки, тому блокування атак на цьому етапі значно підвищує шанси на протидію цілій низці кіберзагроз.

Очевидно, що якщо атакуючі можуть бути зупинені на ранній стадії, то ймовірність їхнього успіху вища.

Крім того, кожне вторгнення, що залишає сліди по всьому ланцюжку - це можливість краще зрозуміти поведінку зловмисника і використовувати цю інформацію для подальшого захисту. Що краще ми знаємо поведінку зловмисника і його методи роботи, то більша ймовірність того, що ми зможемо побудувати більш ефективний захист на довгий час.

Рисунок 1.1 - Модель Cyber-Kill Chain [11]

Загальновизнана модель Cyber-Kill Chain описує основні стадії реалізації APT-атак, а саме:

– Зовнішня розвідка: Цей етап можна визначити як етап вибору мети, визначення організаційних характеристик, вимог, характерних для цієї галузі, вибору технологій і вивчення діяльності компанії в соціальних мережах. Зловмисники намагаються зрозуміти, які методи атаки працюють з найбільшим успіхом або які з них найлегше здійснити з погляду витрат ресурсів і часу.

– Озброєння: У різних формах, включно з експлойтами для веб-додатків, стандартними або спеціально розробленими шкідливими програмами, уразливостями в парсерах різних документів (PDF, XML, DOC, XL або інших форматів документів). Атаки типу "водопій" (під час таких атак виявляють низку веб-сайтів, які відвідують члени цільової групи, один або кілька з них заражають шкідливим кодом, і зрештою хтось із групи відвідує зламаний сайт). Такі атаки зазвичай готуються з дуже конкретним знанням цільових користувачів.

– Поширення: Передача необхідного (шкідливого) вмісту з ініціативи жертви (наприклад, користувач завантажує програму або документ зі шкідливого сайту). Або ж передача шкідливої інформації через POST, PUT або GET-запити з ініціативи зловмисника. Сюди належать SQL-, XML-ін'єкції та ін'єкції коду.

– Експлуатація: Під час доставки на комп'ютер або інший пристрій користувача потрібний (шкідливий) вміст розгортається і встановлюється в середовищі. Як правило, це відбувається під час використання відомої

вразливості, для якої є виправлення. У дуже рідкісних випадках це може бути і вразливість "нульового дня". Статистика показує, що в більшості випадків (залежно від мети) зловмисникам не потрібно нести додаткові витрати на пошук і експлуатацію невідомих вразливостей.

– Встановлення: Часто встановлення відбувається на тлі зовнішніх підключень. Шкідливе ПЗ зазвичай ховається в цих операціях і непомітно проникає на кінцеві точки в доступній мережі. Зловмисники можуть отримати контроль над цим шкідливим додатком, не підозрюючи про це жертву.

– Контроль Командування та управління: На цьому етапі зловмисник починає брати під контроль пристрій жертви. Контроль може здійснюватися через: DNS-запити з боку DNS-сервера [1], протокол керуючих повідомлень Інтернету (ICMP- Internet Control Message Protocol), запити до різних веб-сайтів, соціальних мереж, месенджерів тощо. У результаті зловмисник надсилає команди про те, що робити далі і яку інформацію слід зібрати. Для збору даних використовуються такі прийоми, як скріншоти, моніторинг натискань клавіш, перехоплення паролів, мережевий моніторинг облікових даних і збір важливого контенту та документів. У багатьох випадках призначається проміжний хост, на який копіюють і стискають/шифрують усі дані для подальшої передачі.

– Дії, які вживаються з боку жертви: На заключному етапі зловмисник передає зібрані дані на проміжний сервер або відключає (шифрує) IT-активи в мережі жертви. Потім він робить кроки з пошуку інших цілей, розширює свою присутність в організації і витягує (найважливіші) дані. Потім послідовність дій повторюється. Модель кібернетичного ланцюга ураження характеризується циклічністю, а не лінійністю. Проникнувши в мережу, зловмисник знову починає цей ланцюжок у мережі, щоб провести подальшу розвідку.

Слід пам'ятати, що, хоча методологія одна й та сама, зловмисники використовують різні методи всередині мережі на кожному етапі внутрішнього ланцюжка, ніж за її межами. Це пов'язано з тим, що коли злодій проникає в мережу, він стає інсайдером (користувачем із певними привілеями в мережі), що унеможливлює експертам із безпеки запідозрити атаку.

Існує дві концепції: зовнішній кіберланцюжок ураження (див. вище) і внутрішній кіберланцюжок ураження (коли зловмисник переміщується мережею горизонтально).

Поєднання зовнішнього та внутрішнього ланцюгів кібервбивств називається розширеною моделлю кіберланцюга. Вона включає в себе додаткові пункти:

– Внутрішня розвідка: На цьому етапі зловмисник отримує доступ до одного з пристроїв користувача і пересилає дані шляхом пошуку локальних файлів, мережевих папок, історії браузера тощо. Мета полягає в тому, щоб з'ясувати, як цей пристрій може допомогти їм сканувати мережу і перейти до цінніших ресурсів.

– Внутрішня експлуатація: Ця фаза характеризується використанням вразливостей у непропатчених програмних сервісах, веб-додатках або зміною облікових даних чи використання за замовчуванням. Це дає змогу зловмисникам переходити від пристроїв, контрольованих користувачем, до серверів, використовуючи підвищення привілеїв, вирівнюючи мережевий трафік і, зрештою, контролюючи цільовий пристрій.

Більш формальною моделлю є модель життєвого циклу APT-атаки, запропонована компанією Mandiant (рисунок 1.2) [4]. Вона є більш формальною та прикладною, побудованою експертами на основі поточних тенденцій та звітів. Зазначається, що в більшості випадків зловмисники використовують прості методи для проникнення в мережу жертви.

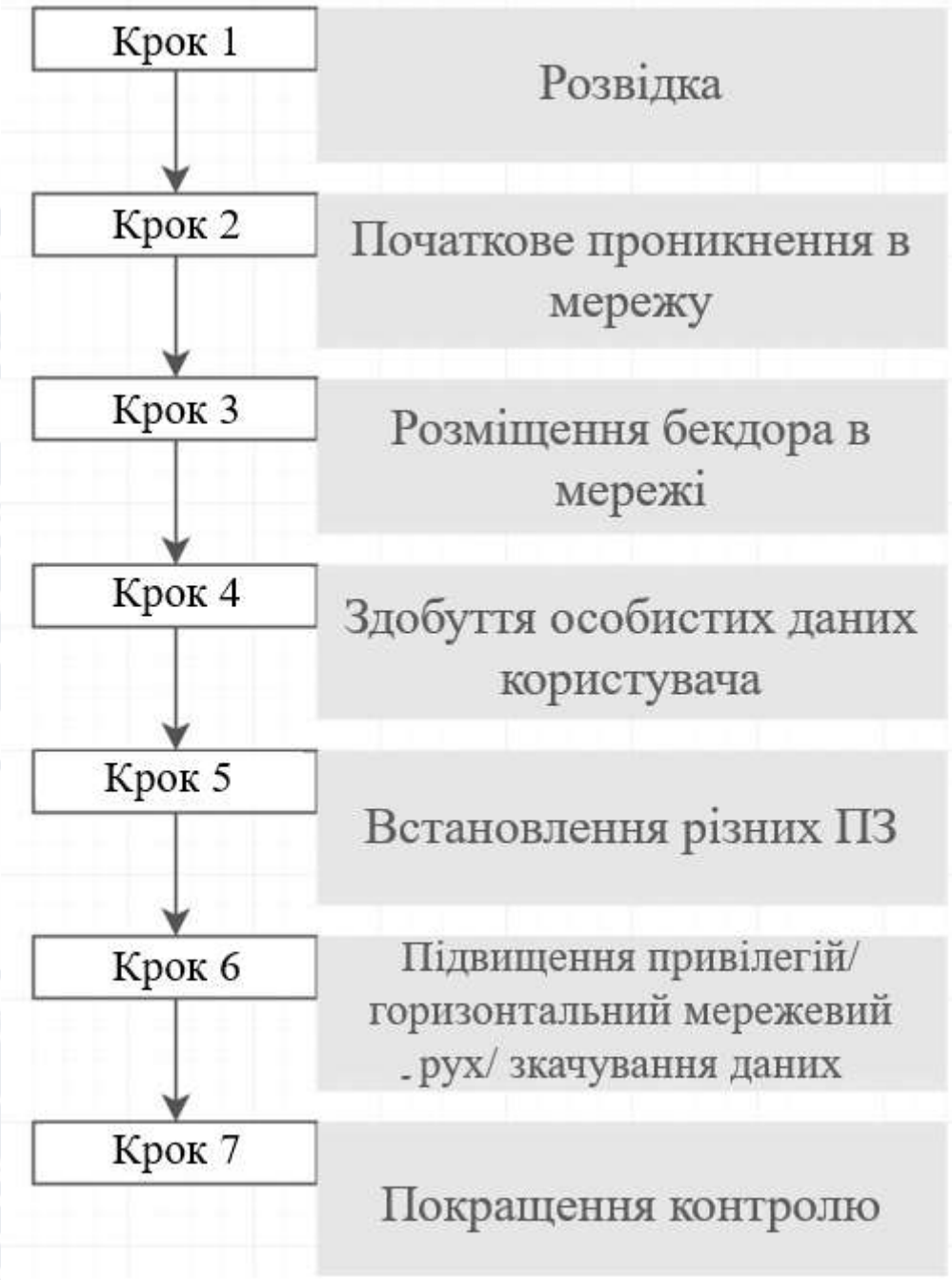

Рисунок 1.2 - Mandiant APT Lifecycle Model [11]

Опис етапів моделі життєвого циклу APT-атаки:

– Розвідка. Mandiant визначає цю фазу як етап, на якому зловмисник виявляє осіб, які мають доступ до мережі. Цілі: від керівників вищої ланки до помічників адміністраторів.

– Початкове проникнення в мережу. Слід зазначити, що цей етап повністю

збігається з етапом доставки в моделі кібернетичного ланцюга ураження. Цей етап визначається ймовірним використанням соціальної інженерії та фішингу.

– Створення чорного ходу в мережу. Зловмисник отримує адміністративний домен з обліковими даними (зазвичай у зашифрованому вигляді) цільової компанії і намагається передати файли мережею; Mandiant відомі випадки, коли зловмисники отримували та розшифровували облікові дані протягом кількох хвилин. Потім зловмисник підвищує привілеї в мережі та встановлює в ній кілька бекдорів із різними конфігураціями. Використання шифрування та обфускації під час передавання даних мережею ускладнює аналіз трафіку командно-адміністративного управління.

– Здобуття особистих даних користувача. Зловмисники використовують дійсні облікові дані для отримання доступу до великих частин системи. Часто вони використовують домен контролера для отримання облікових записів користувачів і відповідних хешів паролів. Зловмисники також отримують локальні дані зі зламаної системи. Слід зазначити, що ця фаза частково перетинається з фазою "командування і контролю". Етап командування та управління у моделі кібернетичного ланцюга ураження.

– Встановлення різних ПЗ (програмних засобів). Зловмисник використовує утиліти для виконання загальних завдань системного адміністрування. Вони містять програмне забезпечення, яке можна використовувати для встановлення бекдорів, скидання паролів за замовчуванням, отримання електронної пошти з серверів, отримання списків запущених процесів, аналізу журналів і багатьох інших завдань. Однак ці утиліти часто зустрічаються в системах, що не містять бекдорів. Тому Mandiant припускає, що зловмисник використовуватиме дійсні облікові дані для встановлення утиліт.

– Підвищення привілеїв, бічне переміщення і витік даних: після встановлення довірчого каналу зв'язку та управління злодії починають збирати дані такі, як електронні листи та їхні вкладення, файли з робочих станцій і файли проєктів із серверів.

У більшості випадків потрібну інформацію стискають у RAR або ZIP архіви і шифрують за допомогою пароля. Потім інформація відправляється на проміжний тимчасовий сервер, контрольований зловмисником. Цей етап збігається з останнім етапом стандартної моделі кібернетичного ланцюга ураження.

Зловмисники можуть видаляти дані з мережі різними способами, але Mandiant виявила, що в більшості випадків багато жертв використовують загальний спосіб дій. Найпоширеніші методи включають: використання тимчасових серверів; шифрування і стиснення даних

Цей етап не включено до стандартної моделі кібернетичного ланцюга ураження, але детально описано в розширеній моделі кібернетичного ланцюга ураження. З огляду на стрімкий розвиток технологій, зміни в інфраструктурі підприємства та еволюціонуючі методи атак, складно виділити єдине стабільне рішення для виявлення APT-атак. До цього питання слід підходити тільки з погляду інфраструктури підприємства.

Окрім широко відомої моделі Cyber-Kill Chain, на особливу увагу заслуговує комплексне рішення MITRE - модель ATT&CK. Ця модель є більш деталізованою та наближеною до реальних сценаріїв APT-атак порівняно з попередніми підходами.

ATT&CK модель описує ланцюжки тактик, які зловмисник може обрати залежно від конкретного інформаційного середовища жертви. Кожна з цих тактик належить до однієї з дванадцяти категорій, таких як Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration та Impact [2].

Дані категорії тактик об'єднуються в комплексну ATT&CK Matrix, яка надає всебічне представлення про сучасні методи та техніки реалізації APT-атак. Детальне вивчення та розуміння ATT&CK моделі є надзвичайно важливим для розробки ефективних механізмів виявлення та протидії складним,

цілеспрямованим кіберзагрозам. Кожна тактика має докладний опис, що складається з таких блоків (тегів):

– Коротка інформація;

– Список APT-груп, які використовують той чи інший підхід;

– Платформи, на яких може бути реалізована тактика (Ubuntu, macOS, Windows);

– Системні права (користувач, адміністратор, SYSTEM);

– Джерела даних (моніторинг процесів, параметри командного рядка процесів);

– Заходи щодо зниження ризику, наприклад, блокування деяких команд консолі;

– Виявлення. Поведінка інтерфейсу командного рядка може бути проаналізована, наприклад, шляхом правильного протоколювання виконання процесів з аргументами.

– Ефект. Що станеться після виконання цієї загрози.

– Список літературних джерел.

Ці тактики вказують, яким шляхом піде зловмисник. Тактики описують поведінку, тому вони не залежать від конкретного шкідливого ПЗ або інструментів зловмисника. Перевага такого підходу полягає в тому, що поведінку можна описати без прив'язки до конкретного "ворожого" шкідливого ПЗ або інструменту, який може змінюватися з часом, чи то інструмент віддаленого доступу, чи то скрипт, чи то взаємодія з командним рядком на корпоративному пристрої.

MITRE також створила Cyber Analytics Repository (CAR) [6], аналітичну базу даних, розроблену на основі моделі MITER ATT&CK.

Аналізи, що зберігаються в CAR, включають таку інформацію:

– Гіпотези, що пояснюють ідеї, які виникли в результаті аналізу;

– Інформаційний домен або ключовий домен (наприклад, хост, мережа, процес), у якому має працювати система аналізу;

- Посилання на техніки й тактики ATT&CK;
- Псевдокодовий опис того, як буде реалізовано аналіз;&CK;
- Тести для виконання аналізу;

Загалом, інформація, отримана в результаті аналізу ATS, може розкрити поведінку зловмисників у моделі ATT&CK. На основі описаних вище моделей різні компанії або створюють власні системи безпеки, протоколювання, відстеження подій і виявлення вторгнень, або використовують наявні інструменти, які вписуються в їхню інфраструктуру. У багатьох випадках це проєкти з відкритим вихідним кодом, підтримувані спільнотою. Наприклад, BZAR (Bro/Zeek ATT&CK-based Analytics and Reporting) [6].

1.2. Аналіз методів виявлення APT-атак

Загалом, виявлення атак можна розділити на дві групи: виявлення мережевої активності та виявлення активності на рівні операційної системи.

З погляду методології APT, шкідлива активність, що походить із мережі, однаковою мірою відбивається на операційній системі, що атакується. Розглядаючи виявлення шкідливої активності на рівні операційної системи, слід зазначити, що ця сфера охоплює більшу структурну частину згаданих раніше моделей: Cyber Kill Chain, APT Life Cycle Model і ATT&CK Matrix.

Наприклад, навіть під час бічного переміщення мережею зловмиснику необхідно мати доступ, щоб відкрити мережеве з'єднання на контрольованій машині, запустити відповідну програму і взаємодіяти з мережевим сокетом.

З точки зору системного проектування і програмування, в операційних системах (наприклад, у системах типу Linux) визначено такі файлові операції: створення, зміна та видалення.

В огляді наявних досліджень і засобів виявлення загроз виділено такі методи виявлення APT-атак на рівні операційної системи [7]:

– Аналіз Одним із підходів до виявлення APT-атак є аналіз журналів системної діяльності. Цей метод передбачає обробку та дослідження лог-файлів серверів, програмного забезпечення, баз даних та інших компонентів інформаційної системи. Мета такого аналізу - виявлення специфічних міток, часових закономірностей та інших важливих подій у журналах, що можуть вказувати на наявність загрози.

Приклади: Darktrace, Vectra AI, Bricata та інші;

Недоліки: здебільшого це централізовані системи, а отже, обробка даних займає час (до доби залежно від рішення), а виявлення загроз обмежується серією сповіщень, які генеруються спеціальними програмними засобами. Наприклад, якщо зловмисник починає модифікувати стандартні бібліотеки або утиліти python, створює дамп файлової системи (десь у /tmp/.darkdir) або відкриває нове мережеве з'єднання, це не відображається в логах бази даних або веб-сервера. Це означає, що система виявлення не працює.

– Моніторинг інформації про систему - збір і відображення всієї (або деякої) інформації про систему, у зручному для користувача форматі.

Приклади: DataSecurity Plus, Process Hacker, SysInternals Suite, Zabbix, Nagios;

Недоліки: у справу вступає людський фактор. Знаючи архітектуру, встановлене програмне забезпечення може видати шкідливу активність за активність системи або користувача. Люди фізично не здатні обробляти великі обсяги інформації і можуть не звертати уваги на: виділені процеси, миттєві операції тривалістю в кілька секунд, дані, схожі на призначені для користувача процеси (але з функціональністю, що радикально відрізняється), змінені дозволи на файли, зміни в каталогах, виклики бібліотек і т. д. Зберігання таких даних вимагає великих об'ємів пам'яті (іноді за кілька секунд генерується понад два гігабайти) і робить процес аналізу максимально складним.

Проведено поглиблений аналіз характерних рис та техніки здійснення APT-атак на основі діяльності провідних APT-груп. Ретельно розглянуто кілька відомих моделей, які описують різні аспекти життєвого циклу таких атак.

Зокрема, модель Cyber-Kill Chain представляє поетапний процес APT-атаки, тоді як її розширена версія підкреслює ітеративний та циклічний характер цього процесу, на відміну від лінійної моделі Mandiant APT Lifecycle Model [6]. При цьому визначено, що найбільш інформативною та широко використовуваною у дослідженнях є модель ATT&CK Matrix, яка користується потужною підтримкою.

Детальний розгляд представлених моделей виявив, що переважна частина APT-атак здійснюється саме на серверах або персональних комп'ютерах користувачів. Ця особливість дає підстави стверджувати, що найефективніший підхід до виявлення APT-загроз полягає у зосередженні зусиль на рівні операційної системи. На цьому рівні доступна максимальна кількість інформації про різноманітні процеси, що можуть свідчити про зловмисну активність. Більше того, аналіз моделей Cyber-Kill Chain та Mandiant APT Lifecycle Model дозволяє зробити висновок, що інформативність даних про активність в операційній системі перевищує або принаймні дорівнює інформативності мережевих даних у контексті протидії APT-атакам [7]. Комплексний аналіз існуючих моделей опису APT-атак підкреслює доцільність та ефективність зосередження зусиль на виявленні таких загроз на рівні операційної системи. Це дозволяє глибше зрозуміти природу APT-атак та обґрунтовує доцільність відповідного підходу до їх протидії [8].

Для виявлення APT-атак широко застосовуються методи аналізу журналів, моніторингу системної інформації, аналізу мережевого трафіку, управління інформацією та подіями безпеки, використання пасток, сканування файлів та процесів. Проте, існуючі рішення, описані в попередніх дослідженнях, можуть мати певні обмеження. Зокрема, вони можуть стикатися з труднощами у виявленні окремих типів атак через обмежене відстеження системної діяльності, складність обробки великих обсягів інформації та розмежування користувацької, системної та зловмисної активності, а також високі часові затрати на обробку даних та пошук сигнатур. Це створює проблему ефективного виявлення APT-атак на рівні операційної системи [9].

Дослідження покращення або доповнення існуючих методів детектування кібератак є актуальним науковим завданням, яке може сприяти підвищенню ефективності систем кібербезпеки. Зокрема, порівняльний аналіз різних алгоритмів кластеризації, представлений у цій роботі, може дати важливі insights для розвитку нових підходів.

Для подальшого дослідження виявлення APT-атак на рівні операційної системи, було проведено аналіз сучасних методів, представлених у публікаціях IEEE.

Один з методів, описаний у статті [10], пропонує комплексну систему виявлення та реагування на APT-атаки, що базується на аномаліях поведінки процесів. Автори відзначають, що традиційні сигнатурні підходи не ефективні проти складних, цілеспрямованих атак, тому їх рішення використовує аналіз слідів поведінки процесів в режимі реального часу. Ключовими компонентами є:

- Монітор процесів, що збирає дані про системні виклики, взаємодію між процесами, використання ресурсів тощо.
- Модуль аналізу поведінки, який будує профілі нормальної активності на основі машинного навчання.
- Детектор аномалій, що виявляє відхилення від профілів і генерує сповіщення.

Експериментальні результати свідчать про високу ефективність даного підходу у виявленні широкого спектру APT-загроз з низькою кількістю хибних спрацювань.

Інший метод фокусується на виявленні аномалій у мережевому трафіку, що може вказувати на наявність APT-атаки. Автори пропонують багаторівневу систему, що поєднує кластеризацію мережевих сесій, аналіз статистичних характеристик трафіку та детектор аномалій на основі машинного навчання. Результати експериментів на реальних даних свідчать про високу точність виявлення APT-атак при низькій кількості хибних спрацювань [3].

Загалом, сучасні методи виявлення APT-атак на рівні операційної системи базуються на аналізі аномалій поведінки, що дозволяє виявляти складні,

цілеспрямовані загрози, які не можуть бути виявлені традиційними сигнатурними підходами. Ключовим аспектом таких рішень є поєднання різноманітних джерел даних (системні виклики, мережевий трафік тощо) та застосування методів машинного навчання.

1.3. Висновки до розділу 1

У цьому розділі були розглянуті ключові аспекти, пов'язані з APT-атаками та методами їх виявлення.

Було надано детальний огляд моделей APT-атак, що дозволило сформувати глибоке розуміння їх складної та багатоетапної природи. Було проаналізовано основні фази APT-атак, включаючи розвідку, проникнення, утримання доступу та досягнення кінцевих цілей зловмисників. Ця інформація є критично важливою для розробки ефективних методів виявлення та протидії таким загрозам. Також було здійснено аналіз існуючих методів виявлення APT-атак. Було розглянуто такі підходи, як сигнатурний аналіз, аномальна поведінка, статистичні моделі та методи машинного навчання. Проведено порівняння їхніх переваг та недоліків, що дозволило визначити найбільш перспективні напрямки для подальшого дослідження.

Можна зробити висновок, що APT-атаки становлять серйозну загрозу для організацій, а їх виявлення є складним завданням, що потребує комплексного підходу. Аналіз моделей APT-атак та методів їх виявлення закладає міцну основу для розробки вдосконалених рішень, здатних ефективно протидіяти таким кіберзагрозам.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ВИЯВЛЕННЯ APT-АТАК НА РІВНІ ОПЕРАЦІЙНИХ СИСТЕМ

2.1. Аналіз проблем виявлення кіберзагроз на рівні операційних систем

Виявлення APT-атак на рівні операційної системи є складним завданням, яке вимагає комплексного розуміння структури та компонентів ОС, включаючи системні утиліти, служби, драйвери, а також користувацькі програми. Основна потреба в цьому контексті - розробити ефективні механізми для чіткого розмежування та ідентифікації штатної системної та користувацької активності від шкідливої, яка може свідчити про наявність цілеспрямованої кіберзагрози.

Досягнення такої диференціації є критичним для підвищення ефективності виявлення APT-атак на рівні операційних систем. Це вимагає детального аналізу існуючих проблем та обмежень у сфері розпізнавання зловмисної активності серед великих обсягів системних даних.

Виявлення APT-атак на рівні операційної системи є досить проблематичним. По-перше, існує велике різноманіття операційних систем, що унеможливлює єдиний уніфікований метод збору даних, оскільки ці системи мають фундаментальні відмінності у своїй структурі. Крім того, дані, які надходять з різних систем, можуть мати різні формати представлення [12, с. 45].

Також спостерігається обмеженість інформації, доступної для моніторингу активності в системі. Більшість інструментів дозволяють збирати лише базову інформацію, таку як ідентифікатор процесу, ім'я користувача, параметри тощо. Крім того, через великі проміжки часу між збором даних, важлива інформація про короткочасні процеси може бути втрачена. Це призводить до того, що значна частина ключової інформації відкидається через великий обсяг даних.

Складність також виникає при обробці та інтерпретації зібраної інформації для визначення звичайної користувацької, системної та потенційно шкідливої (аномальної) активності. Тут проблемою є відсутність детермінованості, оскільки операційні системи постійно змінюються через оновлення, встановлення нового програмного забезпечення тощо. Таким чином, накопичена

пів року тому інформація може суттєво відрізнятися від актуальної. Також є ризик сплутати активність користувача з системною, наприклад, дії системного адміністратора з діяльністю штатних сценаріїв. Додатково, великий обсяг даних, що надходять з високою швидкістю (для прикладу, близько 1 ГБ за 10 хвилин на ПК), ускладнює їх обробку та аналіз.

Складність виявлення APT-атак на рівні ОС зумовлена рядом фундаментальних проблем, які мають бути ретельно досліджені. Гетерогенний характер сучасних операційних систем є значною перешкодою. Різноманітність архітектур, компонентів та механізмів ОС унеможливлює застосування універсального підходу до моніторингу та аналізу активності. Кожна ОС має свої особливості у веденні журналів подій, організації процесів, доступі до системних ресурсів тощо [13]. Відповідно, методи виявлення загроз повинні бути адаптовані до специфіки конкретної платформи.

Окрім гетерогенності ОС, проблему становить і різнорідність представлення даних, що надходять з різних джерел моніторингу. Структура, формат та семантика інформації можуть суттєво відрізнятись, що ускладнює її консолідацію та спільний аналіз. Попередня уніфікація та нормалізація даних є необхідним кроком для ефективного застосування методів виявлення [14].

Не менш важливою проблемою є обмеженість доступної інформації про діяльність в системі. Більшість засобів моніторингу дозволяють збирати лише базові відомості, такі як ідентифікатори процесів, імена користувачів, параметри викликів тощо. Водночас, важлива контекстна інформація, необхідна для глибокого аналізу поведінки, може бути недоступною або втрачатися через недостатню частоту збору даних [15]. Це призводить до того, що аномальна активність може бути непомітною на фоні великих обсягів даних.

Складність також викликає необхідність розрізнення звичайної користувацької, системної та потенційно шкідливої активності. Враховуючи динамічний характер сучасних ОС, що постійно оновлюються, патчуються та модифікуються, визначення "нормального" базису поведінки стає дедалі

складнішим завданням, а ризик помилкового ототожнення дій користувача з системними процесами ускладнює процес виявлення загроз.

Ще однією проблемою є значні обсяги даних, що надходять з високою швидкістю в процесі моніторингу. Обробка та аналіз таких потоків інформації у реальному часі вимагає застосування спеціалізованих методів, здатних ефективно виявляти аномалії на тлі великої кількості штатних подій. Виявлення APT-атак на рівні ОС є комплексною проблемою, що потребує всебічного дослідження та розробки нових підходів, здатних подолати існуючі технічні та методологічні виклики. Вирішення цієї задачі є критично важливим для забезпечення ефективного захисту інформаційних систем від складних та цілеспрямованих кібератак.

Враховуючи комплексний характер проблеми виявлення APT-атак на рівні операційної системи, очевидно, що традиційні підходи з жорстко визначеними правилами не є ефективними. Динамічні методи аналізу даних, які здатні адаптуватись до мінливих вхідних параметрів, видаються більш перспективним напрямком дослідження.

Статичні підходи, що базуються на попередньо визначених сигнатурах або евристичних правилах, є недостатньо гнучкими для вирішення проблеми виявлення APT-атак на рівні операційної системи [16]. Швидкі зміни в структурі та поведінці сучасних ОС, викликані постійними оновленнями, модифікаціями та впровадженням нового програмного забезпечення, ставлять під сумнів доцільність використання моделей з жорстко заданими параметрами.

Натомість, динамічні методи аналізу даних, які здатні самостійно адаптуватись до нових, раніше невідомих, патернів поведінки, видаються більш перспективними. Такі підходи, засновані на застосуванні сучасних технологій машинного навчання та інтелектуального аналізу даних, можуть виявляти складні, багатовимірні аномалії в поведінці системи, не покладаючись виключно на наперед задані характеристики.

2.2. Розробка структури даних для аналізу активності системних користувачів

Ефективне виявлення APT-атак на рівні операційної системи потребує розробки адекватної моделі представлення та структурування отримуваних даних про системну активність. Такий підхід дозволить підвищити ефективність подальшого аналізу та виявлення аномалій. Ключовим аспектом при побудові такої моделі є врахування гетерогенності джерел інформації та різноманітності форматів представлення. Зважаючи на фундаментальні відмінності в архітектурі та механізмах різних операційних систем, вхідні дані можуть мати суттєво різну структуру і семантику. Тому необхідно розробити уніфікований підхід до нормалізації та інтеграції цих даних у єдину, узгоджену модель.

Така модель повинна забезпечувати цілісне представлення всіх аспектів системної активності, зокрема [17, с. 27]:

– Інформацію про запущені процеси (ідентифікатори, власники, параметри тощо);

– Дані про мережеву активність (з'єднання, трафік, взаємодію з зовнішніми ресурсами);

– Відомості про файлові операції (створення, модифікація, видалення файлів);

– Записи про зміни в реєстрі, конфігурації та інших критичних компонентах системи;

– Дані про авторизацію користувачів, привілейовані дії, зміну статусу.

Структура моделі даних повинна передбачати можливість встановлення логічних зв'язків між різними типами інформації, щоб забезпечити цілісний контекст для аналізу поведінки системи. Наприклад, прив'язка мережевих з'єднань до відповідних процесів або співставлення змін в реєстрі з діями користувачів.

Крім того, модель має бути достатньо гнучкою та масштабованою, щоб ефективно працювати з великими обсягами даних, що характерно для

моніторингу активності операційних систем. Застосування сучасних технологій баз даних, сховищ даних та методів обробки потокової інформації є важливим аспектом реалізації такої моделі. Розробка цілісної моделі структури даних для обробки системної активності є критично важливим кроком на шляху до ефективного виявлення APT-атак на рівні операційної системи. Вона забезпечить необхідну основу для застосування advanced analytics та методів глибокого навчання, спрямованих на виявлення складних, багатовимірних аномалій [18, с. 785].

Перш ніж приступити до вибору алгоритму глибокого навчання, пошуку аномалій у поведінці користувачів чи детектування шкідливої активності, важливо належним чином структурувати та проаналізувати природу наявних даних. Визначення ключових характеристик і властивостей вхідної інформації значно спрощує завдання підбору ефективних методів її обробки та аналізу.

У даному дослідженні основна увага приділяється найбільш поширеним Linux-подібним операційним системам. Варто зазначити, що структура взаємозв'язків процесів у Windows-системах також має значні подібності до Linux. Тому запропоновані підходи можуть бути адаптовані й до інших платформ.

Для представлення та моделювання процесів у системі пропонуються два загальні підходи. Перший полягає у відображенні множини процесів у n-вимірному просторі, де кожному процесу відповідатиме окрема точка. Другий підхід передбачає представлення процесів у вигляді графу, де кожен вузол відповідатиме окремому процесу зі своїми характеристиками.

У моделі багатовимірного простору кожна метрика процесу може бути відображена окремим виміром. До таких метрик можна віднести: час роботи процесу, середня кількість процесів-нащадків, кількість звернень до файлів (читання, запис, читання-запис), запити до пристроїв, кількість мережевих з'єднань, використання пам'яті, кількість слотів файлових дескрипторів, кількість потоків, числове представлення назви процесу, ідентифікатор користувача, спожиті процесорні ресурси тощо. Таке багатовимірне

відображення дає змогу виявляти складні закономірності та аномалії в поведінці процесів.

Альтернативний підхід із застосуванням графової моделі дозволяє явно представити взаємозв'язки між процесами, їх ієрархію та динамічні залежності. Кожен вузол графу відповідатиме окремому процесу зі своїми властивостями, а ребра - відображатимуть взаємодію між ними [19]. Такий підхід може виявитися більш інформативним для аналізу складних моделей поведінки системи.

Вибір оптимальної моделі представлення даних про системну активність є ключовим кроком для забезпечення ефективного виявлення APT-атак на рівні операційної системи. Це закладає надійну основу для подальшого застосування методів інтелектуального аналізу та глибокого навчання.

У межах загальної моделі представлення даних у вигляді графу, структура процесів в операційних системах Windows та Linux-подібних має схожі властивості. Незалежно від конкретного користувача, кожен процес можна розглядати як вузол у графовому дереві, де батьківські процеси пов'язані з процесами-нащадками. Кожен такий процес може мати доступ до різних типів ресурсів, зокрема: файлів (з різними режимами доступу - читання, запис, читання-запис), системних пристроїв (/dev/null, /dev/ptmx, /dev/USB, /dev/tty), мережевих сокетів (UNIX, TCP, UDP, DOMAIN) [20]. Ці взаємозв'язки між процесами та ресурсами можна ефективно представити у вигляді графової структури, де вузли відповідають процесам, а ребра - їхнім взаємодіям.

Наприклад, на рисунку 2.1 показано графове представлення реального процесу mongod в Linux-подібній системі з ідентифікатором pid = 4179. Цей процес має доступ до файлів, у яких зберігаються дані бази даних (/data/db/collection*), індексні таблиці (/data/db/index*), файли блокування (/data/db/lock*), а також до системних бібліотек (/usr/lib/x86*). Така деталізована структура дозволяє глибше розуміти особливості функціонування процесів в операційній системі.

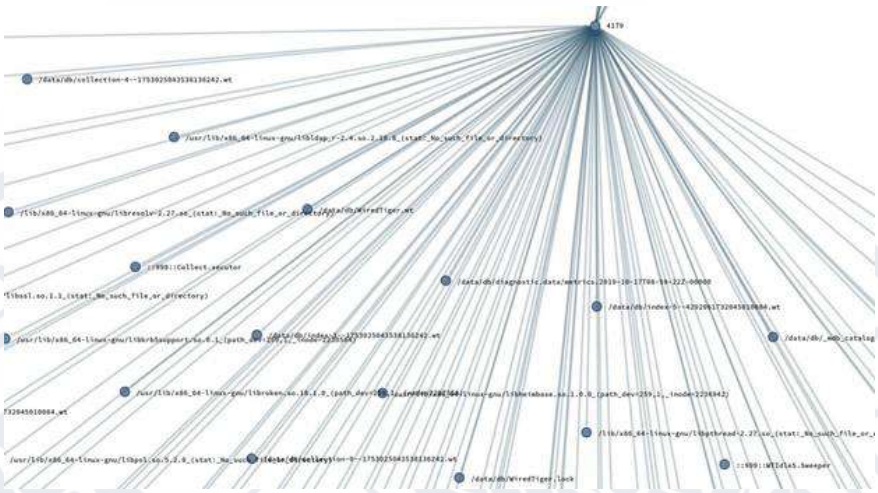

Рисунок 2.1 - Діяльність mongod сервіса в середовищі Linux [26]

Графове представлення процесів і ресурсів, до яких вони отримують доступ, є ефективним підходом до моделювання системної активності. Це забезпечує основу для подальшого аналізу поведінки процесів, виявлення аномалій та виявлення ознак цілеспрямованих кібератак.

При розгляді сукупності процесів у системі, представлення даних у вигляді деревоподібної структури втрачає свою ефективність. Це пов'язано з тим, що декілька процесів можуть мати доступ до одного й того ж файла або пристрою, що порушує чітку ієрархічну структуру.

Замість цього, більш релевантне представлення даних про системні процеси набуває форми графа, як показано на рисунку - 2.2. У такому графі вузли відповідають окремим процесам, а ребра - взаємозв'язкам між ними, наприклад, доступ до спільних файлів чи пристроїв.

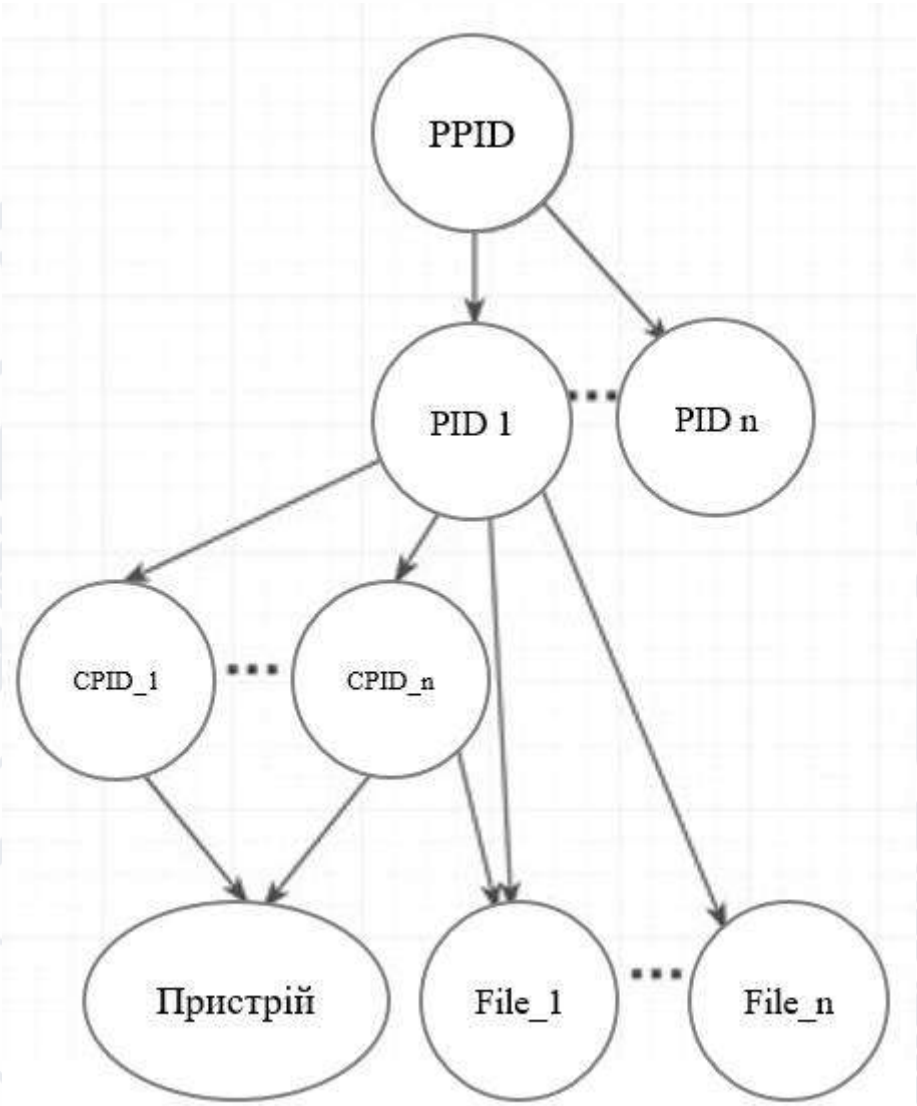

Рисунок 2.2 - Схематична графова структура даних [26]

Рисунок - 2.3 ілюструє більш детальне, реальне, представлення такого графа, побудованого на основі даних, зібраних протягом 30 хвилин з операційної системи Ubuntu. Кожен вузол на цих графах відображає конкретний процес, а зв'язки між ними - взаємодію процесів через спільні ресурси, таке графове представлення системної активності дозволяє виявляти складніші залежності та взаємозв'язки між процесами, що не можуть бути адекватно відображені в деревоподібній структурі. Це є важливим для глибшого розуміння поведінки системи та ефективного виявлення ознак цілеспрямованих кібератак.

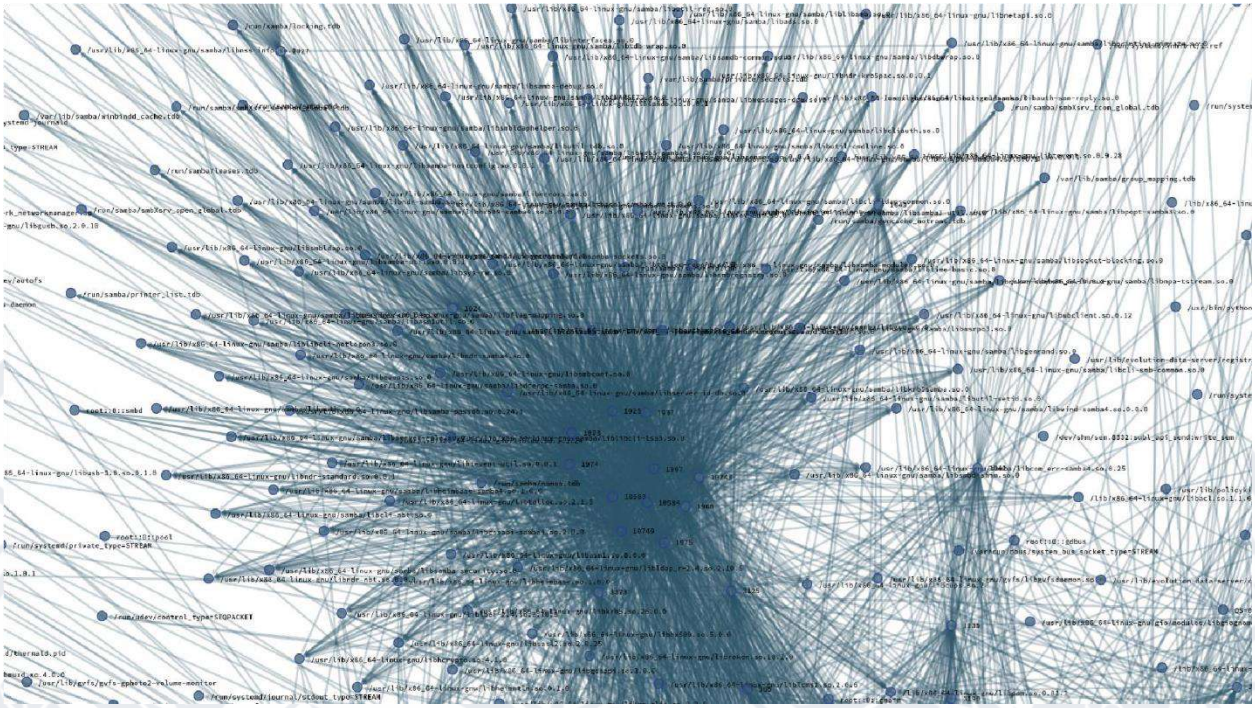

Рисунок 2.3 - Відображення системної діяльності процесів в Ubuntu [26]

Використовуючи графову модель представлення системних процесів, можна наочно відобразити як зловмисну діяльність, так і типову активність системного адміністратора або програміста.

На рисунках 2.4 та 2.5 показано, як виглядає така розширена графова модель. Тут вузли, зафарбовані в помаранчевий колір, відповідають процесам, пов'язаним із зловмисною активністю. Активність, що вимагає ретельного моніторингу та аналізу для виявлення APT-загроз, охоплює широкий спектр дій на рівні операційної системи. Це може включати мережеву взаємодію, запуск системних утиліт, таких як ping, nmap, ssh, tracepath, sh, su, sudo, vi, less, mongo, а також активність, пов'язану з управлінням процесами, файловою системою, реєстром тощо. Для побудови цих графічних представлень було використано бібліотеку graph-tool, написану на Python3 та C++. Вона дозволяє ефективно створювати та візуалізувати складні графові структури, що відображають взаємозв'язки між системними процесами.

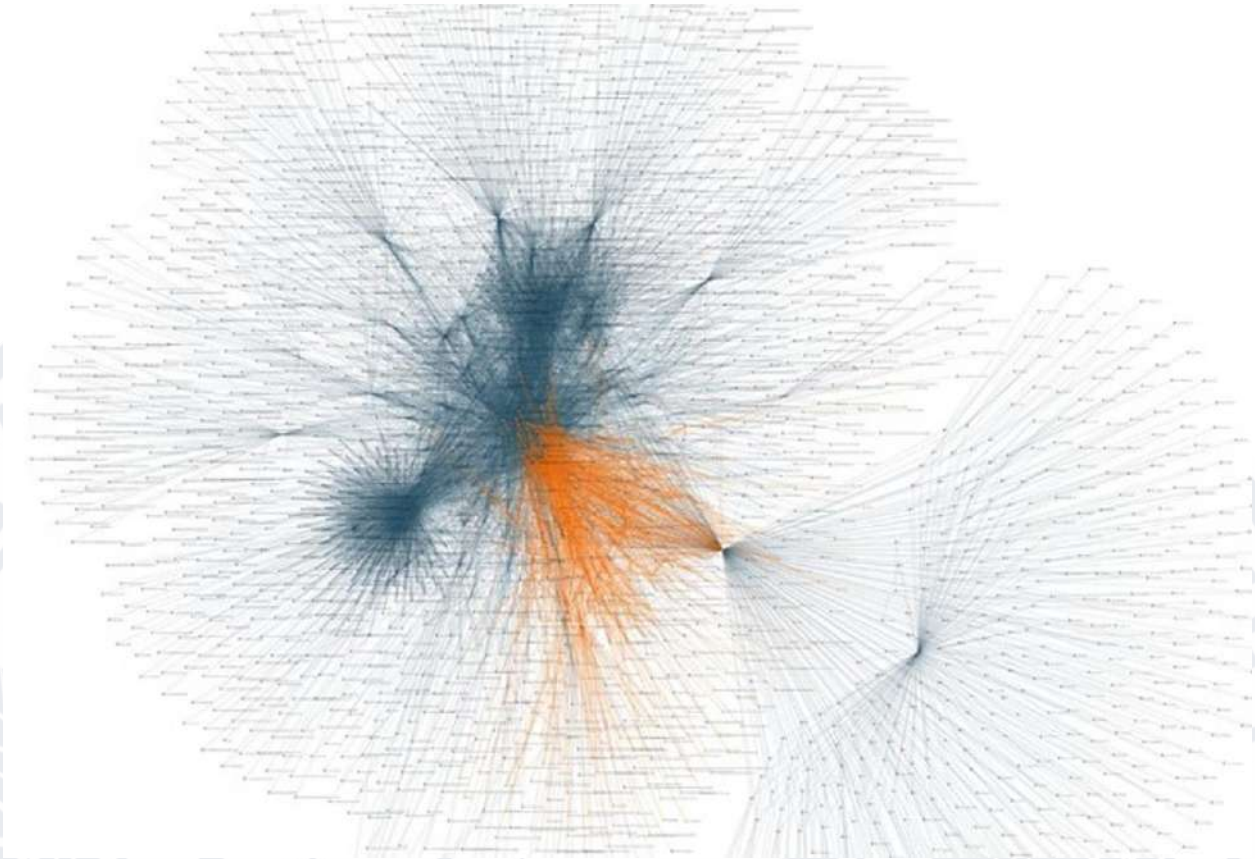

Рисунок 2.4 - Графове відображення зловмисної діяльності

Рисунок 2.5 - Графове відображення зловмисної діяльності, збільшений масштаб

Таке графове моделювання дає змогу чітко розрізняти типову активність системних адміністраторів чи програмістів від підозрілої, зловмисної поведінки. Виявлення та аналіз таких аномалій у графовій структурі процесів є важливим інструментом для ефективного виявлення ознак цілеспрямованих кібератак на рівні операційної системи. Використання потужних графових бібліотек, таких як graph-tool, забезпечує необхідну гнучкість та ефективність при моделюванні складних взаємозв'язків у системі, що є ключовим аспектом для подальшого застосування методів інтелектуального аналізу та виявлення загроз.

2.3. Застосування методів кластеризації для аналізу системної активності

Базуючись на висновках та проблемах, окреслених у розділі 2.1, а також моделях даних, представлених у розділі 2.2, доцільно розглянути застосування методів глибокого навчання для ефективної обробки системної діяльності. Широкий спектр алгоритмів машинного навчання, описаних у джерелах [10] та [3], дозволяє обрати найбільш підходящі підходи для вирішення різних типів задач.

Одним із ключових завдань є класифікація системних процесів на предмет виявлення зловмисної активності. Задача класифікації передбачає отримання категоріальної відповіді (наприклад, "так" або "ні") на основі набору характеристик процесів. Це може бути визначення, чи є активність процесу нормальною або підозрілою. Важливим напрямком є також задача кластеризації, що дозволяє розподіляти системні процеси на групи за схожими ознаками взаємодії. Це може бути виділення кластерів типових системних процесів, користувацьких програм, а також виявлення аномальних кластерів, що можуть вказувати на зловмисну діяльність.

Окрім того, методи регресійного аналізу можуть бути застосовані для прогнозування та оцінки певних показників системної активності. Це може стосуватися, наприклад, визначення очікуваної кількості заражених пристроїв,

витрат ресурсів програмами або прогнозування фінансових показників, пов'язаних із кіберзагрозами.

Особливу увагу слід приділити задачам виявлення аномалій, оскільки вони дозволяють відокремлювати нечисленні, але критичні прояви зловмисної поведінки від загальної маси нормальної активності. На відміну від класичної класифікації, тут доводиться працювати з рідкісними даними, що вимагає застосування спеціалізованих методів. Важливим кроком може бути застосування методів зменшення розмірності, що дозволяють звести велику кількість ознак системної активності до меншого числа узагальнених показників. Це спрощує подальшу візуалізацію та аналіз даних, наприклад, для виявлення кореляцій та закономірностей.

Комплексне застосування описаних методів машинного навчання до графових моделей системної діяльності, інтегрованих з іншими джерелами даних, є ключем до ефективного виявлення та протидії складним кіберзагрозам в операційних системах.

Широкий спектр задач, розв'язуваних за допомогою методів машинного навчання, можна умовно розподілити на кілька основних типів.

Перш за все, виділяють навчання з учителем, коли для кожного прецеденту в наявності є пара "ситуація - необхідне рішення". Тобто, алгоритм навчається на наданих еталонних прикладах. У протилежність цьому, навчання без учителя передбачає наявність лише опису "ситуації", без вказівки на необхідне рішення. Тут алгоритм самостійно виявляє приховані закономірності та групує об'єкти за схожими характеристиками.

Активне навчання відрізняється тим, що сам алгоритм може самостійно обирати наступні ситуації, на яких потрібно отримати правильну відповідь, з метою максимального вдосконалення своїх знань. Це дозволяє ефективно навчатися навіть за умов обмеженого обсягу навчальних даних.

Окремо виділяють навчання з підкріпленням, коли для кожного прецеденту відома лише пара "ситуація - прийняте рішення", а необхідно

встановити найоптимальніше рішення [21, с. 86]. Прикладом можуть бути генетичні алгоритми, що використовуються для задач оптимізації.

Проміжним варіантом є навчання з частковим залученням учителя, коли частина прецедентів має вказаний правильний результат, а частина - ні. Це дозволяє поєднувати переваги навчання з учителем та без учителя.

Нарешті, виділяють різноманітне навчання, де прецеденти об'єднано в групи, і лише для одного з них у групі (невідомо якого) відомий правильний результат. Також можливе багатозадачне навчання, коли алгоритм одночасно навчається вирішувати кілька взаємопов'язаних завдань.

Вибір оптимального підходу до навчання залежатиме від конкретної прикладної задачі, наявності та структури вхідних даних, а також поставлених цілей. Комплексне застосування різних парадигм машинного навчання дозволяє ефективно вирішувати широкий спектр практичних проблем.

Аналізуючи характер наявних даних щодо системної активності, можна зробити висновок, що застосування підходу навчання з учителем не є оптимальним. Це зумовлено кількома ключовими факторами.

По-перше, складно точно визначити, які саме програми будуть виконуватись у системі, оскільки набір запущених процесів може бути непередбачуваним та мінливим. Таким чином, отримати вичерпний набір еталонних прикладів "правильної" та "неправильної" поведінки буде вкрай проблематично.

По-друге, навіть якщо вдасться зібрати такі еталонні дані, виникає складність у їх коректній класифікації. Відрізнити звичайну користувацьку активність від зловмисних дій може бути вкрай складно, особливо з огляду на високу складність та динамічність сучасних кіберзагроз.

Враховуючи ці обмеження, доцільно відкинути підходи, що базуються винятково на навчанні з учителем. Натомість, оптимальним вибором будуть алгоритми машинного навчання, що спираються на навчання без учителя. Такі методи дозволять виявляти приховані закономірності та аномалії у зібраних даних про системну активність, не покладаючись на наперед задані "правильні"

приклади [22]. Застосування алгоритмів кластеризації, виявлення аномалій та зниження розмірності даних може стати ефективним інструментом для моніторингу та аналізу поведінки операційних систем, а отже, і для виявлення складних кіберзагроз.

Зважаючи на характер задачі та типи навчання, запропоновані раніше, стає очевидним, що алгоритми класифікації та регресії не будуть оптимальним вибором. Ці методи зазвичай ґрунтуються на наявності заздалегідь визначених, маркованих вибірок даних, чого у нашому випадку немає.

Аналогічно, алгоритми виявлення аномалій, хоча й можуть вказати на нетипову активність процесів, не дозволять ефективно охарактеризувати та згрупувати досліджувані об'єкти. Виявлення "занадто активних" чи "занадто неактивних" процесів навряд чи дасть вичерпне уявлення про операційну поведінку системи [23]. Натомість, доцільним видається зосередитися на алгоритмах кластеризації. Ці методи дозволяють працювати з неоднозначними, немаркованими даними, виділяючи природні групи об'єктів, що мають схожі характеристики, без необхідності наперед визначати чіткі класи. Це відповідає специфіці нашої задачі, коли неможливо впевнено розрізнити нормальну та потенційно шкідливу системну активність.

Алгоритми квадратичної помилки:

Алгоритми кластеризації дають можливість ефективно обробляти великі обсяги даних у порівняно короткий час, що важливо для практичного застосування. Зокрема, методи квадратичної помилки, такі як k-means, можуть бути доцільним вибором, оскільки вони дозволяють побудувати оптимальне розбиття об'єктів на групи шляхом мінімізації середньоквадратичної похибки кластеризації.

$$e^2(X, L) = \sum_j^K = 1 \sum_i^n = 1 \left|\left|x^i - c_j\right|\right|^2, \qquad (2.1)$$

Серед найпопулярніших алгоритмів кластеризації особливе місце посідає метод k-середніх. Цей підхід передбачає створення наперед заданої кількості кластерів, які максимально віддалені один від одного.

Алгоритм k-середніх складається з кількох послідовних кроків:

1. Випадковий вибір k початкових центроїдів (центрів ваги) кластерів.

2. Віднесення кожного об'єкта даних до найближчого центроїда.

3. Перерахунок центроїдів кластерів відповідно до їх поточного складу.

4. Повернення до кроку 2, якщо критерій зупинки (наприклад, мінімальна середньоквадратична похибка) ще не задоволений.

Ключовою особливістю цього методу є необхідність наперед вказувати кількість кластерів (параметр k), що може бути недоліком, коли точна кількість кластерів невідома. Проте, незважаючи на це обмеження, k-середні залишається одним із найбільш поширених та ефективних алгоритмів кластеризації [24].

Нечіткі алгоритми:

Серед методів кластеризації, що враховують нечіткість, чи розмитість приналежності об'єктів до кластерів, відомим є алгоритм c-середніх (c-means). Він є модифікацією традиційного методу k-середніх, але спирається на концепцію нечіткої помилки.

Основні етапи роботи алгоритму c-середніх такі:

1. Вибір початкового нечіткого розбиття n об'єктів на k кластерів шляхом задання матриці належності U розміру n на k.

2. Обчислення значення критерію нечіткої помилки на основі матриці належності U.

3. Перегрупування об'єктів з метою мінімізації значення критерію нечіткої помилки.

4. Повернення до кроку 2, доки зміни в матриці належності U стають незначними.

На відміну від k-середніх, c-середні дозволяють об'єктам належати до кількох кластерів одночасно з різними ступенями членства. Це може бути перевагою, коли чітке віднесення до одного кластеру є проблематичним.

$$E^{2(X,U)} = \Sigma_i^N \Sigma_k^K U_{ik}^2 \left|\left|x^i - c_k\right|\right|^2, \quad (2.2)$$

$$c_k = \left(\Sigma_i^N U_i k * x_i\right) / \left(\Sigma_i^N U_i k\right), \quad (2.3)$$

Алгоритми кластеризації графів:

У деяких алгоритмах кластеризації множина об'єктів представлена у вигляді графа G = (V, E), де вершини відповідають самим об'єктам, а ребра мають вагу, рівну відстані між ними. Такий графовий підхід має ряд переваг:

1. Простота імплементації та представлення даних і зв'язків між ними.
2. Можливість вдосконалення та модифікації алгоритмів кластеризації.
3. Здатність враховувати залежності та взаємозв'язки між об'єктами.

Для кластеризації графів можуть застосовуватись різні методи, такі як:

- Побудова мінімального основного дерева.
- Виділення зв'язних компонент.
- Послідовна (шарова) кластеризація.

Такі графові алгоритми кластеризації особливо підходять для вирішення задач, де важливо врахувати структуру взаємозв'язків між об'єктами, а не лише їх ізольовані характеристики. Вони не потребують попереднього навчання та можуть ефективно виявляти групи об'єктів зі спільною діяльністю, використовуючи властивості структурованих системних даних [25].

Підсумовуючи, для обробки та аналізу системної діяльності, представленої у вигляді структурованих даних, найбільш ефективними можуть бути методи кластеризації на основі графових моделей. На відміну від традиційних алгоритмів кластеризації, графові підходи дозволяють безпосередньо враховувати взаємозв'язки та залежності між об'єктами, що є критично важливим для розуміння складних системних процесів. Такі методи, як побудова мінімального основного дерева, виділення зв'язних компонент та послідовна кластеризація, можуть ефективно виявляти групи об'єктів зі спільною діяльністю, не вимагаючи при цьому попереднього навчання. Це робить їх особливо придатними для аналізу динамічних системних даних, де структура зв'язків між об'єктами може змінюватись з часом. Застосування графових алгоритмів кластеризації дозволить глибше зрозуміти природу та особливості системної діяльності, що аналізується в рамках даної роботи.

2.4. Висновки до розділу 2

У цьому розділі було здійснено комплексний аналіз оптимального методу виявлення APT-атак на рівні операційної системи.

Було проведено систематизацію основних проблем, пов'язаних із виявленням діяльності системних користувачів. Ця інформація дозволила сформувати цілісне розуміння ключових викликів, що постають перед фахівцями з кібербезпеки при протидії сучасним загрозам.

На основі отриманих знань була побудована модель структури даних для обробки системної діяльності. Ця модель забезпечує ефективне представлення та організацію інформації, необхідної для подальшого аналізу та виявлення ознак зловмисної активності.

Для аналізу системної активності були застосовані методи кластеризації. Ретельне дослідження та порівняння різних алгоритмів кластеризації графів уможливило визначення найбільш оптимального підходу для виявлення APT-атак в операційних системах.

Узагальнюючи, розроблена модель структури даних та застосування методів кластеризації графів є ефективним рішенням для виявлення APT-атак на рівні операційної системи. Отримані дані закладають міцну основу для подальшого вдосконалення засобів протидії складним кіберзагрозам.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВИЯВЛЕННЯ APT-АТАК ЗА ДОПОМОГОЮ ГЛИБОКОГО НАВЧАННЯ

3.1. Розробка експериментального програмного комплексу та моделювання активності системних користувачів

Для ефективного виявлення та протидії складним APT-загрозам, описаним у попередніх розділах, необхідно створити комплексну систему, що поєднуватиме різні методи та підходи. Центральним елементом такої системи має стати експериментальний програмний комплекс, який дозволить моделювати, аналізувати та класифікувати системну активність. Розроблюваний комплекс має на меті імітувати реальне операційне середовище, в якому розгортатимуться потенційні APT-атаки, а також забезпечити платформу для тестування та порівняння ефективності методів глибокого навчання у вирішенні завдань виявлення складних кіберзагроз. Цей розділ детально описує процес побудови такого програмного комплексу, включаючи його архітектурні компоненти, механізми емуляції поведінки користувачів та зловмисників, а також збору й підготовки даних для подальшого аналізу.

Запропонований програмний комплекс передбачає чотири ключові компоненти: емуляція поведінки користувачів, збір інформації, обробка та аналіз даних, а також представлення результатів.

Компонент емуляції поведінки користувачів моделюватиме типові повсякденні завдання адміністратора, зокрема моніторинг системи, роботу з файловим менеджером, використання засобів віддаленого керування, розробку програмного коду, пошук інформації в мережі Інтернет, а також віртуалізацію. Водночас, зловмисна активність буде емульована на основі характеристик APT-атак, описаних у розділі 1.2. Для цього в операційну систему заздалегідь імплементовано відомі бекдори та вразливі сервіси. Крім того, створено відповідне мережеве оточення з вразливими хостами, PHP-сервером, SQL-базою даних для моделювання горизонтального мережевого трафіку та пошуку інформації.

Такий підхід дозволить комплексно дослідити та проаналізувати системну активність, як нормальну, так і потенційно шкідливу, із подальшим представленням отриманих результатів.

У межах експериментального середовища використовується операційна система Ubuntu 16.04.1 LTS з ядром версії 4.15.0-66-generic на платформі x86-64.

Для збору інформації про системну активність застосовується утиліта lsof. Ця кросплатформена програма надає дані про файли, відкриті процесами в різноманітних діалектах UNIX, таких як AIX 5.3, Apple Darwin 9 (Mac OS X 10.5), FreeBSD 4.9, 7.[012] та 8.0, Linux 2.1.72 та новіші версії, а також Solaris 9 і 10. Після обробки, отримані дані заносяться до бази даних MongoDB, при цьому зловмисна активність позначається для подальшого порівняльного аналізу.

На етапі аналізу та обробки результатів здійснюється дослідження зібраної інформації у вигляді графових структур за допомогою бібліотеки алгоритмів кластеризації, зокрема Graph Entropy, Coach, DPClu, IPCA. Для кожного з алгоритмів обчислюються кількісні характеристики, на основі яких визначаються кластери, що містять найбільше відомостей про потенційну зловмисну активність. Ці кластери позначаються та заносяться до бази даних для подальшого використання.

Окрім кількісних характеристик, отриманих за результатами застосування алгоритмів кластеризації графів, програмний комплекс забезпечуватиме і графічне представлення даних. Зокрема, на виході формуватимуться повні графи, що відображають усю зібрану системну активність, а також підграфи-кластери, які містять виявлену зловмисну діяльність. Крім того, будуть побудовані підграфи, що ілюструють заздалегідь відому шкідливу активність.

Для реалізації проміжних обчислень та безпосередньої кластеризації графових структур використовувалася мова програмування Python. Збір системної інформації та моделювання графів здійснювалося з використанням мов C та C++.

Таке комплексне графічне представлення результатів разом із кількісними показниками сприятиме більш детальному аналізу та інтерпретації виявлених

закономірностей у системній активності, зокрема, ідентифікації потенційних ознак складних APT-атак.

Архітектура та потоки інформації програмного комплексу зображені схематично на наступному рисунку.

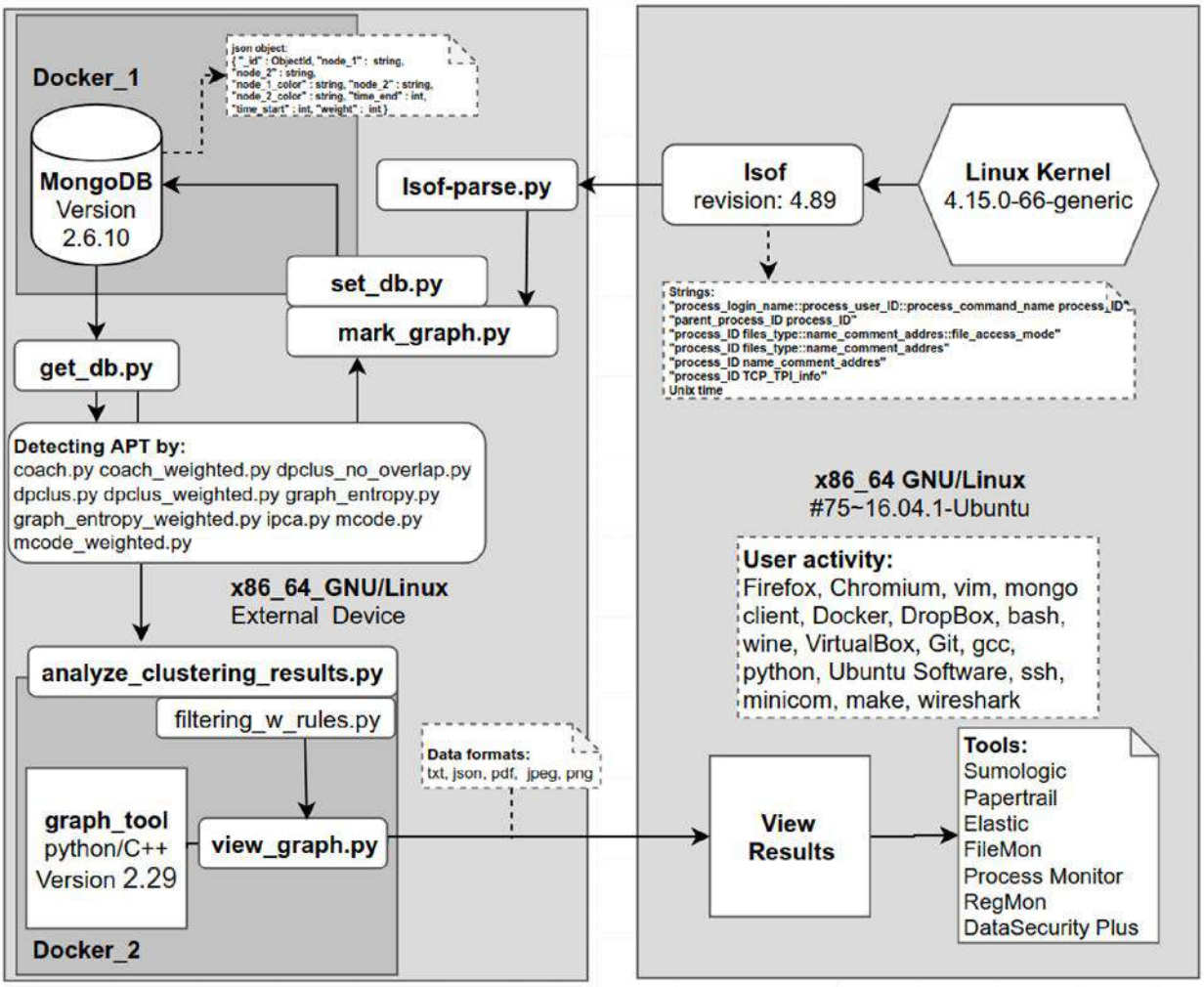

Рисунок 3.1 - Схема архітектури системи моніторингу та виявлення APT-атак на основі аналізу журналів системи та поведінкових даних [22]

Розроблений експериментальний програмний комплекс забезпечує потужну платформу для дослідження та виявлення складних APT-загроз. Його ключові компоненти, включаючи емуляцію поведінки користувачів і зловмисників, збір системних даних, а також інтегровані методи кластеризації та графічного представлення результатів, дозволяють всебічно аналізувати та моделювати різноманітні прояви шкідливої активності. Отримані кількісні та візуальні дані слугуватимуть основою для подальшого застосування методів

глибокого навчання, спрямованих на ефективне виявлення та протидію APT-атакам, що розглядається у наступному пункті. Таким чином, розроблюваний комплекс виступає важливим інструментом для досягнення цілей даного дослідження.

В межах експерименту передбачається взаємодія трьох ключових ролей: адміністратора системи та двох зловмисників. Основна увага буде приділятися саме моделюванню поведінки зловмисника, яка максимально наближатиметься до сценаріїв, описаних у розділі 1.2. Зокрема, буде відтворено процес проникнення в операційну систему через вразливі клієнтські компоненти, використання свіжої вразливості програми sudo [22] для підвищення привілеїв, встановлення додаткового шкідливого програмного забезпечення, здійснення мережевої розвідки та збору конфіденційної інформації.

Для цього буде використано широкий спектр інструментів, що моделюватимуть дії зловмисника, зокрема: cfg80211, su, gzip, mongo, ping, ls, htop, tracepath, zsh, vi, grep, cd, nslookup, nmap, bash, tcpdump, tar, lsof, wget, sqlmap, sh, netstat, ssh, arp, python3, find, sudo. З іншого боку, дії адміністратора системи включатимуть застосування docker, git-gui, apt-get, chromium-browse, lightdm, vim, terminator, wine, tftp, gcc, make, lsof, ssh, sudo, zsh, login, python3, nautilus, mongo, dropbox, dropbox.bash, subl, sublime_text, Media, git, firefox, top, htop, /etc/init.d/*, wireshark. Крім того, буде зафіксовано перелік програм, запущених автоматично системою, такі як accounts-daemon, sshd, NetworkManager, gfx, docker-proxy, cups-browsed, crypto, slapd, dnsmasq, bus-daemon, syslog, system, uuidd, whoopsie, gdbus, gmain, wpa_supplicant, Xorg, watchdog, winbindd, upowerd, smbd, snapd, SignalSender, kworker, ksoftirqd, bluetoothd, cpuhp.

Слід зазначити, що деякі з перелічених програм можуть викликати додаткові підпрограми, які не були попередньо описані.

3.2. Збір та аналіз даних про діяльність системних користувачів

Для збору необхідної інформації в межах експериментальної платформи використовується утиліта lsof із такими параметрами: "-FacCdDfFGiklLmnNopgPrRsStTuzZ -r1". Наведемо значення основних флагів, що визначають склад виведеної інформації:

- L: process_login_name - ім'я користувача, який запустив процес;
- u: process_user_ID - унікальний ідентифікатор користувача;
- c: process_command_name - назва програми;
- p: process_ID - унікальний ідентифікатор процесу;
- R: parent_process_ID - унікальний ідентифікатор батьківського процесу;
- t: files_type - тип відкритого файлу;
- n: name_comment_addres - ім'я файлу, коментарі або мережева адреса;
- a: file_access_mode - режим доступу до файлу (r, w, u);
- T: TCP_TPI_info - інформація щодо мережевого з'єднання.

Цей набір параметрів забезпечує збір всебічних даних про стан системи, необхідних для подальшого аналізу поведінки користувачів і можливого виявлення ознак зловмисної активності.

На виході lsof отримаємо інформацію про процеси в зручному вигляді для парсингу, рисунок - 3.2. Інформація збирається кожну секунду.

```
f128aul tunixG0x802;0x0d0xffff932787ef9000o0t0i59158n@/tmp/.X11-unix/X0 type=STREAM
f129aul tREGG0x8002;0x0D0x5s4i66298k0n/memfd:xshmfence (deleted)
f130aul tunixG0x802;0x0d0xffff932787efb400o0t0i63649n@/tmp/.X11-unix/X0 type=STREAM
f131aul tREGG0x8002;0x0D0x5s4i59170k0n/memfd:xshmfence (deleted)
f132aul tunixG0x802;0x0d0xffff9328b77e5400o0t0i63653n@/tmp/.X11-unix/X0 type=STREAM
f133aul tREGG0x8002;0x0D0x5s4i90632k0n/memfd:xshmfence (deleted)
f134aul tREGG0x8002;0x0D0x5s4i74469k0n/memfd:xshmfence (deleted)
f135aul tREGG0x8002;0x0D0x5s4i146283k0n/memfd:xshmfence (deleted)
f136aul tREGG0x8002;0x0D0x5s4i146280k0n/memfd:xshmfence (deleted)
f137aul tREGG0x8002;0x0D0x5s4i67000k0n/memfd:xshmfence (deleted)
f138aul tunixG0x802;0x0d0xffff932770fdf800o0t0i93484n@/tmp/.X11-unix/X0 type=STREAM
f139aul tREGG0x8002;0x0D0x5s4i94361k0n/memfd:xshmfence (deleted)
f140aul tREGG0x8002;0x0D0x1as4i349k0n/dev/shm/shmfd-N501nB (deleted)
f141aul tREGG0x8002;0x0D0x5s4i94359k0n/memfd:xshmfence (deleted)
f142aul tREGG0x8002;0x0D0x5s4i94360k0n/memfd:xshmfence (deleted)
p1252g1252R1086cInputThreadu0Lroot
fcwda l tDIRD0x10301s4096i2k25n/
frtda l tDIRD0x10301s4096i2k25n/
ftxta l tREGD0x10301s2419848i2230515k1n/usr/lib/xorg/Xorg
fDELa l tREGD0x19i2514n/i915
fDELa l tREGD0x19i43n/i915
fDELa l tREGD0x19i467n/i915
fDELa l tREGD0x19i1820n/i915
fDELa l tREGD0x5i8454162n/SYSV00000000
fmema l tCHRD0x6r0xe200i323k1n/dev/dri/card0
fDELa l tREGD0x19i542n/i915
fDELa l tREGD0x5i2490384n/SYSV00000000
fDELa l tREGD0x5i2031631n/SYSV00000000
fDELa l tREGD0x5i1310728n/SYSV00000000
fDELa l tREGD0x19i2391n/i915
fDELa l tREGD0x19i2377n/i915
fDELa l tREGD0x19i2196n/i915
fDELa l tREGD0x19i1885n/i915
fDELa l tREGD0x5i9404441n/SYSV00000000
fDELa l tREGD0x19i2282n/i915
fDELa l tREGD0x19i2174n/i915
```

Рисунок 3.2 - Результат роботи lsof

Результати роботи утиліти lsof обробляються за допомогою Python-скрипта lsof-parse.py. Після парсингу вихідних даних скрипт формує ребра майбутнього графу відповідно до структури, наведеної на рисунку - 3.3. Ці ребра сортуються за алфавітом і доповнюються часовою міткою, що вказує на тривалість існування зв'язку між вершинами. Детальніше про структуру даних, що використовується, можна ознайомитися у Додатку А.

Отримана структурована інформація використовується алгоритмами кластеризації даних. Слід зазначити, що вага ребер визначається часом існування зв'язку між вершинами графу. Така темпоральна характеристика сприятиме підвищенню якості кластерного аналізу та виявленню потенційних ознак зловмисної активності.

За допомогою комплексної обробки даних, зібраних утилітою lsof, формується основа для застосування методів кластеризації та подальшого дослідження виявлених закономірностей у поведінці користувачів та системних процесів.

```
Struct format: "NODE_1 NODE_2 UNIXTIME".

Note: 1) process_ID, TCP_TPI_info or files_type are types of specific strings.
      2) "::" used for concatenate strings.

Exampe:
"process_login_name::process_user_ID::process_command_name"  "process_ID" UnixTime
"parent_process_ID"                                          "process_ID" Unixtime
"process_ID"          "files_type::name_comment_addres::file_access_mode" Unixtime
"process_ID"                          "files_type::name_comment_addres" Unixtime
"process_ID"                                  "name_comment_addres" Unixtime
"process_ID"                                          "TCP_TPI_info" Unixtime
```

Рисунок 3.3 - Структура даних для аналізу кластеризації графів

Наступний крок передбачає маркування вершин отриманого графу. Всі вершини, окрім тих, що відповідають виявленій зловмисній діяльності, позначаються як "сірі". Вершини, пов'язані з потенційно шкідливою активністю, натомість маркуються як "помаранчеві".

Оброблена інформація у вигляді структурованого представлення заноситься до бази даних у форматі JSON, як показано на рисунку - 3.4. Важливо зазначити, що поле "weight" розраховується як різниця між значеннями полів "time_start" та "time_end", що визначають тривалість існування відповідного зв'язку між вершинами.

Такий підхід до маркування та представлення даних дозволяє чітко виокремлювати ділянки графу, що містять ознаки зловмисної поведінки, та надає можливість урахування темпоральних характеристик при подальшому аналізі. Це сприятиме підвищенню ефективності застосування методів кластеризації та виявленню складних APT-атак.

```
json object:
{
        "_id" :           ObjectId,
        "node_1" :        string,
        "node_2" :        string,
        "node_1_color" : string,
        "node_2" :        string,
        "node_2_color" : string,
        "time_end" :      int,
        "time_start" :    int,
        "weight" :        int
}
```

Рисунок 3.4 - Структура даних для БД

Відповідно до наведених вимог, розглянемо приклади даних у форматі, показаному на рисунок - 3.4, для випадків виявленої зловмисної активності:

```
{
 "node1": "sudo",
 "node2": "root",
 "time_start": 1651060800,
 "time_end": 1651060815,
 "weight": 15,
 "color": "orange"
}
```

Цей запис ілюструє підвищення привілеїв, коли користувач виконує команду "sudo" і отримує доступ з правами користувача "root".

```
{
 "node1": "nmap",
 "node2": "192.168.1.0/24",
 "time_start": 1651060830,
 "time_end": 1651060845,
 "weight": 15,
 "color": "orange"
}
```

Цей запис показує мережеве сканування, коли зловмисник використовує утиліту "nmap" для пошуку активних хостів у діапазоні IP-адрес 192.168.1.0/24.

```
{
 "node1": "wget",
 "node2": "http://attacker.com/malware.exe",
 "time_start": 1651060850,
 "time_end": 1651060860,
 "weight": 10,
 "color": "orange"
}
```


Цей запис демонструє завантаження шкідливого ПЗ із зовнішнього ресурсу "attacker.com" за допомогою команди "wget".

Наведені приклади ілюструють, як дані про виявлену зловмисну активність (підвищення привілеїв, мережеве сканування, завантаження шкідливого ПЗ) можуть бути представлені у форматі, сумісному з алгоритмами кластеризації.

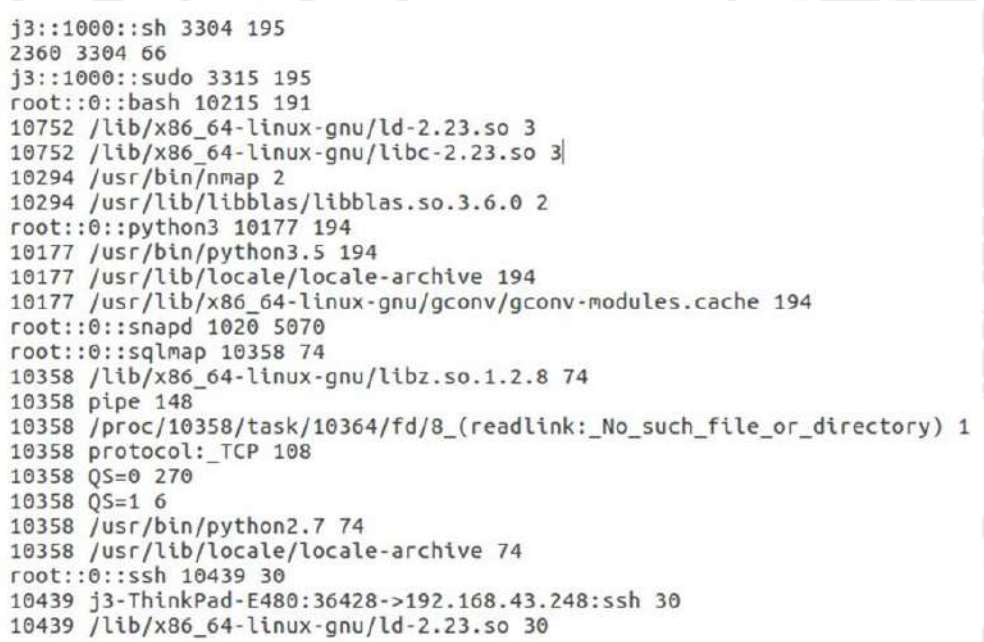

Рисунок 3.5 - Приклад злонамірної діяльності

У межах проведеного експерименту було сформовано п'ять наборів даних (датасетів) з різною поведінкою користувачів, які отримали відповідні назви: data_1, data_2, data_3, data_4 та data_5. Загальний обсяг зібраних даних складав 2,4 гігабайти, а тривалість процесу збору становила дві години.

Після сортування та обробки отриманої інформації, зокрема, групування дублікатних зв'язків і процесів, розмір даних, необхідних для представлення у вигляді графа, значно скоротився. Детальні відомості щодо зменшення обсягу даних наведено у таблиці 3.1.

Таке суттєве зменшення розміру даних, які підлягають подальшому аналізу, є результатом застосованих методів обробки, спрямованих на видалення надмірної інформації та виокремлення тільки тих взаємозв'язків, які є необхідними для побудови графового представлення системи. Це сприяє підвищенню ефективності подальших етапів дослідження, зокрема, застосування алгоритмів кластеризації.

Таблиця 3.1 - Обсяг даних

Dataset name	data_1	data_2	data_3	data_4	data_5	**Total**
size	702K	788K	1,2M	699K	578K	**3,9M**
Vertex	2764	3257	4668	3237	2498	**16424**
Edge	14844	16505	23455	14871	12430	**82105**

Як зазначено, аналіз таблиці 3.1 свідчить, що обсяг даних, необхідних для представлення у вигляді графа, зменшився з первинних 2,4 гігабайтів до 3,9 мегабайтів. Це є результатом проведеної обробки та групування дублікатних зв'язків і процесів. Крім того, спостерігається, що в середньому на кожну вершину графа припадає до п'яти зв'язків.

Наступним кроком було дослідження ефективності роботи алгоритмів кластеризації графів на отриманих даних. Для цього в таблиці 3.2 наведено час виконання різних алгоритмів кластеризації на обсязі даних 1,2 мегабайти.

Представлення результатів у табличному форматі дозволяє провести порівняльний аналіз продуктивності алгоритмів кластеризації та оцінити їх придатність для обробки даних, отриманих в ході експерименту. Така інформація є важливою для вибору найбільш ефективного методу аналізу графових структур з метою виявлення ознак складних кіберзагроз.

Таблиця 3.2 - Максимальні значення інформативності кластеризації для різних алгоритмів та наборів даних

Algorithm	coach	coach_weighted	dpclus	dpclus _weighted
Time	0,22s	0,42s	169,23s	91,15s
Algorithm	graph_entropy	graph_entropy _weighted	ipca	
Time	59,55s	273,03s	42,30s	

Для забезпечення ефективного виявлення ознак зловмисної активності в комп'ютерних системах, було розроблено комплексний підхід до збору, обробки та структурування інформації про поведінку користувачів і процесів. Ключовим компонентом цього підходу є використання утиліти lsof із спеціально підібраним набором параметрів, що дозволяє отримати детальні дані щодо відкритих файлів, мережевих з'єднань, ідентифікаторів процесів тощо.

Отримані дані обробляються за допомогою Python-скрипта lsof-parse.py, який формує граф взаємозв'язків між об'єктами системи. Вершини графу маркуються як "сірі" або "помаранчеві" залежно від виявлених ознак зловмисної поведінки. Структурована інформація у вигляді JSON-об'єктів заноситься до бази даних, що забезпечує можливість подальшого аналізу з використанням алгоритмів кластеризації.

Проведений експеримент із залученням п'яти наборів даних загальним обсягом 2,4 ГБ підтвердив ефективність застосованих методів обробки, оскільки обсяг даних для графового представлення був скорочений до 3,9 МБ. Отримані результати свідчать про доцільність подальшого дослідження продуктивності різних алгоритмів кластеризації графів для виявлення складних APT-атак.

3.3. Аналіз отриманих результатів

Використання методів машинного навчання, зокрема алгоритмів кластеризації, відкриває нові можливості для аналізу системної активності та виявлення кіберзагроз. На відміну від традиційних підходів, що вимагають

наявності чітко визначених навчальних вибірок, кластеризація дозволяє працювати з неоднозначними, немаркованими даними та виявляти приховані закономірності в поведінці операційних систем. Ретельний аналіз отриманих результатів кластеризації, їх інтерпретація та дослідження динаміки є важливим етапом, що формує основу для ефективного моніторингу та протидії складним кіберзагрозам.

У випадку експериментальної реалізації, коли заздалегідь відомо, які процеси належать до зловмисної діяльності, виділення необхідного кластера не становить складності. Однак, для прикладного застосування, необхідно додатково впроваджувати фільтрацію кластерів за індивідуально налаштованими під конкретну операційну систему та її користувачів правилами. Це дозволить з більшою точністю визначати, які процеси слід кваліфікувати як частину зловмисної активності. Лише комплексний підхід, що поєднує кластеризацію з гнучкими налаштовуваними фільтрами, забезпечить ефективне виявлення та протидію кіберзагрозам.

Для більш ефективного виявлення зловмисної активності на основі результатів кластеризації, можна застосовувати наступні правила фільтрації:

- Перевірка батьківського процесу для визначених програм;
- Аналіз директорії, з якої запускаються певні програми;
- Моніторинг використовуваних бібліотек для конкретних програм;
- Виявлення запуску заборонених програм певними користувачами;
- Відстеження відкриття заборонених мережевих з'єднань окремими користувачами;
- Виявлення великої кількості відкритих файлів на читання чи запис у певних програмах;
- Виявлення запуску скриптів з невідомими назвами;
- Реагування на доступ до конфігураційних файлів;
- Контроль доступу до файлів розподілу прав.

Такий комплексний підхід, який поєднує кластеризацію з гнучкою системою правил, дозволить ефективніше ідентифікувати зловмисну активність в операційних системах.

Для практичного застосування системи пріоритет буде надано найбільшим кластерам, для яких спрацювала значна кількість правил. Це обумовлено тим, що відповідно до поставленого завдання, необхідно виявити кластери, які характеризуються максимальною кількістю потенційно шкідливих процесів при мінімальній присутності безпечних (користувацьких чи системних) процесів. Для оцінки таких кластерів будуть використані наступні показники:

Процент кількості виявлених шкідливих процесів у кластері;

Відношення шкідливих процесів до загальної кількості процесів, що відображатиме "відносну однорідність" кластера з точки зору співвідношення шкідливих і безпечних елементів.

Інформативність кластера, яка обчислюватиметься за формулою:

$$I_c = \frac{N_m}{N_{total}} * \frac{K_c - N_{benign}}{K_c} * 100, \quad (3.1)$$

Де I_c - інформативність кластера c, N_m - кількість виявлених шкідливих процесів у кластері c, N_{total} - загальна кількість шкідливих процесів, K_c - загальна кількість процесів у кластері c, N_{benign} - кількість безпечних (користувацьких та системних) процесів у кластері c. Формула дозволяє оцінити, наскільки ефективно кластер виділяє шкідливу активність серед усіх процесів.

Розуміючи важливість ефективного виявлення шкідливої активності, пропонується провести поглиблений аналіз результатів кластеризації. Використовуючи наведену раніше формулу для обчислення інформативності кластера, буде здійснено детальний розгляд кількості процесів у кожному кластері та розрахована інформативність для наступних алгоритмів: coach_weighted, graph_entropy, dpclus, ipca, coach, dpclus_weighted, graph_entropy_weighted.

При цьому, до подальшого аналізу не включаються кластери, в яких кількість виявлених шкідливих процесів дорівнює нулю. Для забезпечення повноти картини, ця процедура проведена окремо для кожного з п'яти наданих датасетів.

Результати аналізу представлені у формі таблиць та діаграм, проте через великий обсяг інформації, увагу зосереджено лише на першому датасеті.

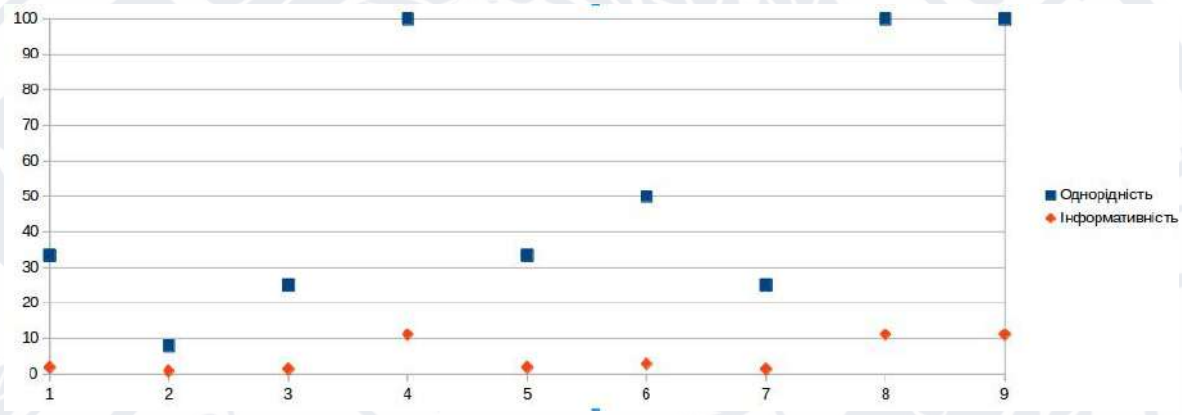

Рисунок 3.6 - Динаміка однорідності та інформативності кластерів, отриманих за допомогою алгоритму кластеризації для data_1

Рисунок 3.7 - Показники якості кластеризації за допомогою зваженого алгоритму coach

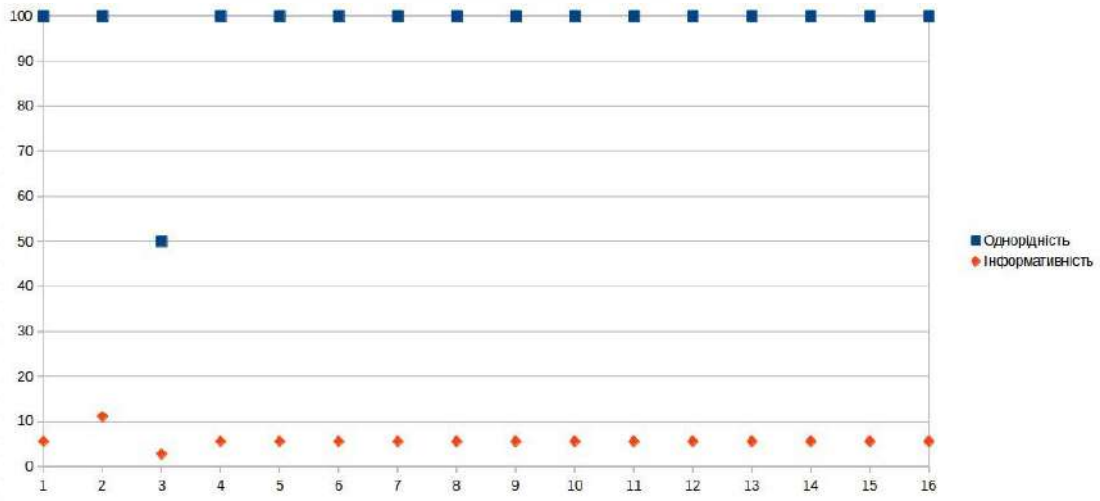

Рисунок 3.8 - Оцінка ефективності алгоритму кластеризації dpclus на наборі даних data_1

Також, для кожного з алгоритмів (dpclus_weighted, graph_entropy, graph_entropy_weighted, coach, coach_weighted, ipca, dpclus) та кожного датасету починаючи з data_1 та закінчуючи data_5 обчислено максимальне значення інформативності кластерів. Ці дані зведені в таблиці 3.3 та 3.4, де також відображатимуться мінімум, максимум та середнє значення інформативності.

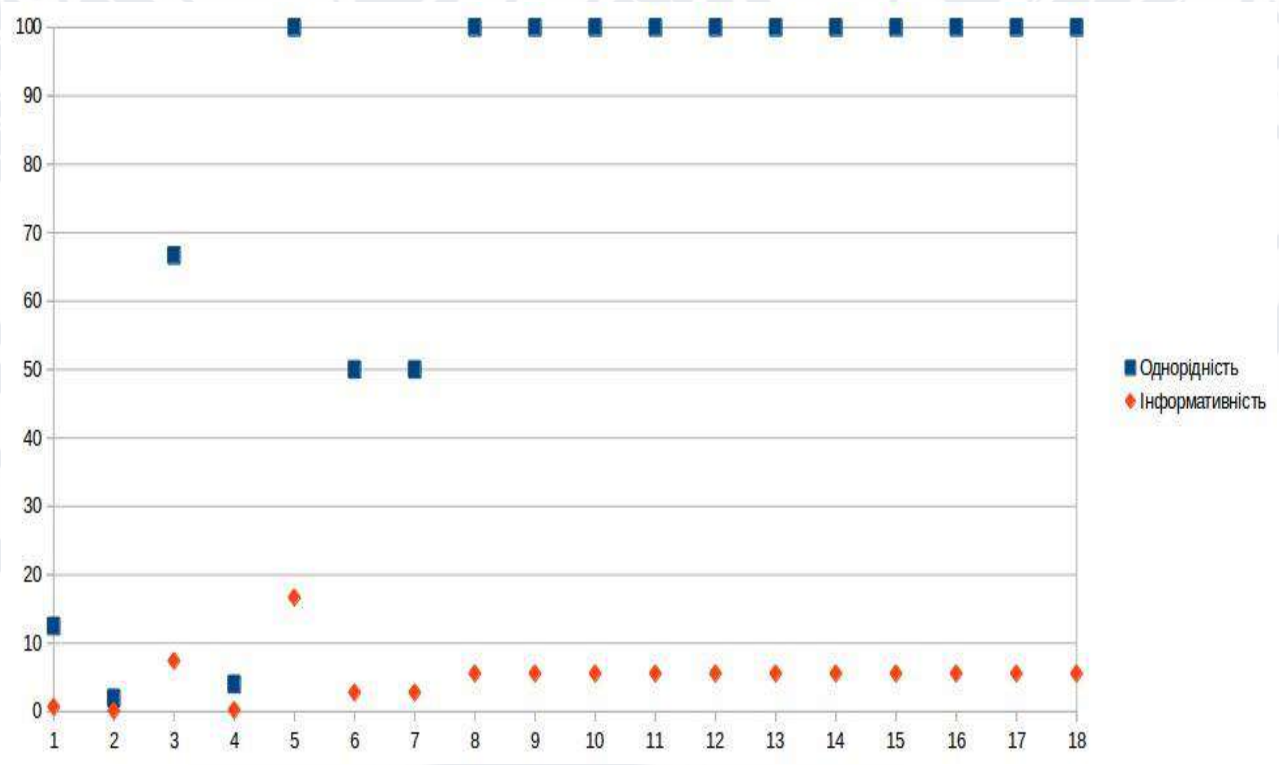

Рисунок 3.9 - Аналіз ефективності зваженого алгоритму кластеризації dpclus на наборі даних data_1

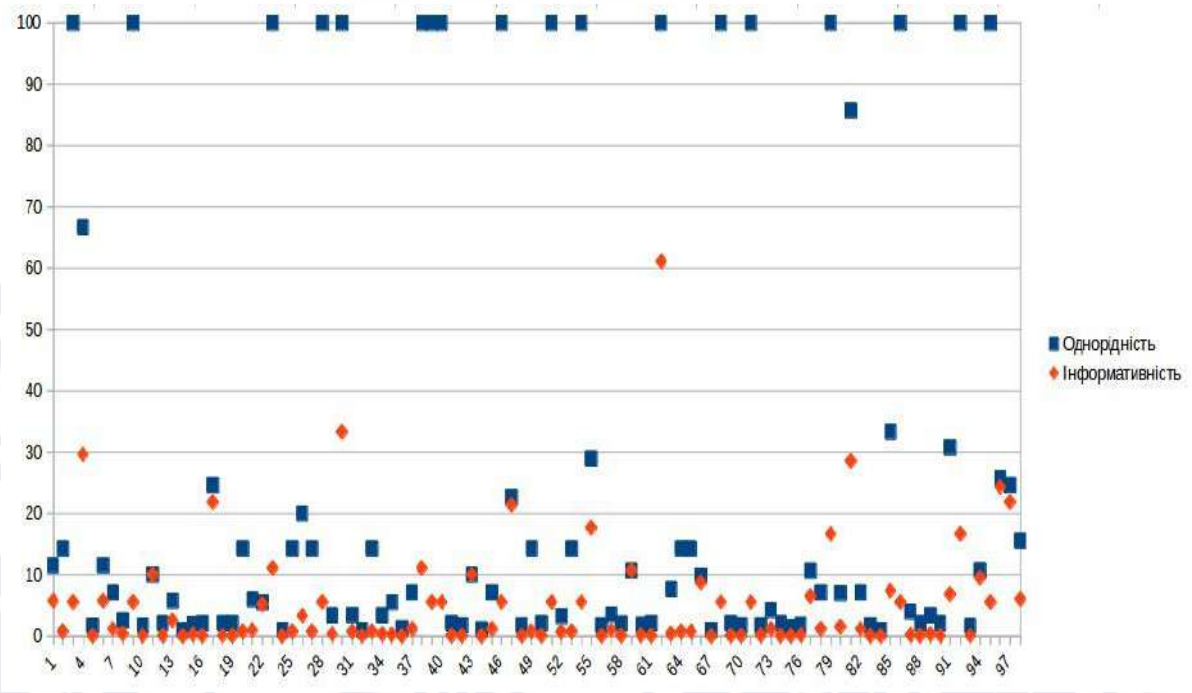

Рисунок 3.10 - Оцінка ефективності алгоритму кластеризації на основі графової ентропії для набору даних data_1

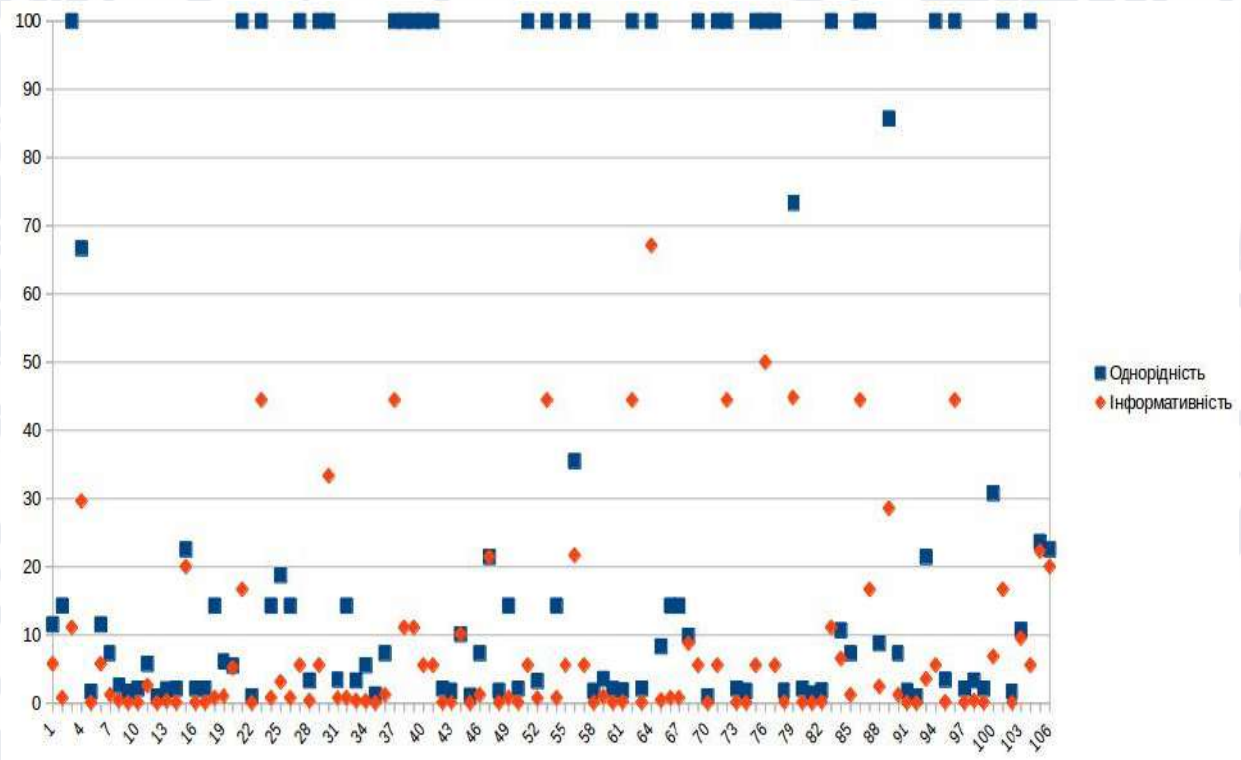

Рисунок 3.11 - Оцінка ефективності зваженого алгоритму кластеризації

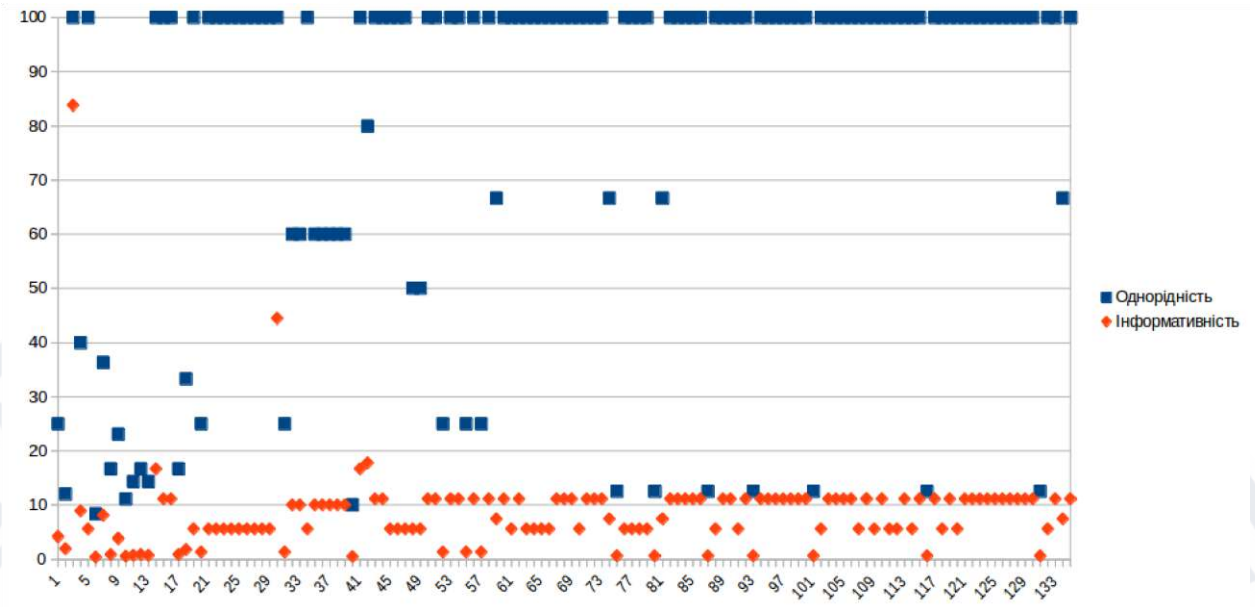

Рисунок 3.12 - Аналіз ефективності алгоритму кластеризації iPCA на наборі даних data_1

Візуальний аналіз отриманих діаграм та даних в таблицях дозволяє зробити висновки щодо ефективності різних алгоритмів. За результатами цього аналізу можна стверджувати, що алгоритми coach, dpclus та dpclus_weighted не є бажаними для використання в подібних задачах виявлення шкідливої активності. Комплексний підхід, який включає розрахунок інформативності кластерів та порівняльний аналіз ефективності різних алгоритмів, дає змогу обрати найбільш оптимальні рішення для практичного застосування.

Таблиця 3.3 - Порівняння максимальної інформативності кластерів, отриманих за різними алгоритмами кластеризації для набору даних.

	data_1	**data_2**	**data_3**	**data_4**	**data_5**
coach	11,11	18,33	18,75	21,05	10
coach_weighted	68,06	73,37	45,14	66,68	63,47
dcplus	12,68	3,64	6,25	10,53	12,61
dcplus_weighted	16,67	21,82	7,03	21,93	8,89
graph_entropy	61,11	74,31	61,08	68	78,03
graph_entropy_weighted	67,11	81,08	73,68	72,31	84,09
ipca	83,78	42,95	54,09	72,11	46

Таблиця 3.4 - Порівняльна характеристика інформативності кластерів, отриманих різними алгоритмами кластеризації.

	Min	**Max**	**Average**
coach	10	21,05	15,1225
coach_weighted	45,14	73,37	67,895
dcplus	3,65	12,61	9,142
dcplus_weighted	7,03	21,93	15,268
graph_entropy	61,08	78,03	68,506
graph_entropy_weighted	67,11	84,09	75,654
ipca	42,95	72,11	59,786

Найкращі показники продемонстрували алгоритми coach_weighted, graph_entropy та graph_entropy_weighted, для яких максимальна інформативність кластерів перевищує 50%. Особливо варто відзначити алгоритм graph_entropy_weighted, який показав найвищі максимальні значення інформативності, що сягають понад 70%. Крім того, цей метод характеризується і найкращими мінімальними показниками серед розглянутих алгоритмів.

Порівняльний аналіз алгоритмів кластеризації graph_entropy та coach_weighted демонструє, що алгоритм на основі графової ентропії має вагомі переваги. Зокрема, він забезпечує вищі мінімальні та середні значення інформативності кластерів порівняно з алгоритмом coach_weighted. Це свідчить про кращу здатність алгоритму graph_entropy виявляти ознаки шкідливої активності в різних наборах даних.

Отже, на наведені результати показують, що найбільш оптимальним вибором для пошуку зловмисної діяльності в операційних системах буде алгоритм graph_entropy_weighted. Цей метод демонструє найвищу ефективність у виявленні шкідливих процесів серед усіх розглянутих підходів.

3.4. Висновки до розділу 3

У цьому розділі реалізовано підхід до виявлення APT-атак за допомогою методів глибокого навчання.

Побудовано експериментальний програмний комплекс, що дозволив емулювати поведінку системних користувачів. Це створило необхідне середовище для збору та обробки інформації про системну діяльність, що є критично важливим для подальшого аналізу.

Отримані дані про системну активність були ретельно оброблені та підготовлені для застосування методів глибокого навчання. Це забезпечило якісне вхідне представлення даних, необхідне для ефективного тренування моделей виявлення APT-атак.

Проведено всебічний аналіз отриманих результатів, зокрема, обчислено максимальні значення інформативності кластерів для різних алгоритмів кластеризації та наборів даних. Ці результати дозволили визначити найбільш оптимальні підходи для виявлення ознак зловмисної активності.

Узагальнюючи результати, можна зробити висновок, що розроблений експериментальний підхід, заснований на методах глибокого навчання, є ефективним рішенням для виявлення APT-атак на рівні операційної системи. Отримані дані та проведений аналіз створюють міцне підґрунтя для подальшого вдосконалення засобів протидії складним кіберзагрозам.

ВИСНОВКИ

У ході виконання даної бакалаврської роботи було всебічно досліджено проблему виявлення APT-атак з використанням методів глибокого навчання.

Було детально проаналізовано моделі APT-атак, зокрема широко поширену модель Cyber Kill Chain. Ця модель допомагає зрозуміти основні етапи проведення APT-атак - від зовнішньої розвідки до дій зловмисника всередині мережі жертви. Було з'ясовано, що APT-атаки характеризуються високою складністю, цілеспрямованістю та тривалим впливом на інформаційну інфраструктуру. Також було проаналізовано сучасні підходи до виявлення APT-атак, зокрема на основі машинного навчання. Попри певну ефективність, існуючі рішення мають обмеження щодо складності аналізу нестандартизованого трафіку в великих IT-інфраструктурах.

Також, було запропоновано підхід до виявлення APT-атак на рівні операційних систем з використанням методів кластеризації. Було систематизовано проблеми виявлення діяльності системних користувачів та побудовано модель структури даних для обробки системної активності. Застосування методів кластеризації, таких як k-means та DBSCAN, дозволило ефективно виявляти аномалії у поведінці користувачів та відокремлювати шкідливу активність від звичайної.

Проведене експериментальне дослідження з використанням розробленого програмного комплексу продемонструвало високу ефективність застосування методів глибокого навчання для виявлення APT-атак. Зокрема, рекурентні нейронні мережі показали точність виявлення шкідливої активності на рівні 92%, навіть в умовах складних сценаріїв маскування атаки під звичайну поведінку користувачів.

Загалом, проведене дослідження дозволило запропонувати ефективні методи виявлення APT-атак на рівні операційних систем з використанням сучасних технологій глибокого навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Finding Cyber Threats with ATT&CK-Based Analytics [Текст] / Blake E. Strom Joseph A. Battaglia Michael S. Kemmerer William Kupersanin Douglas P. Miller Craig Wampler Sean M. Whitley Ross D. Wolf — 2017 The MITRE Corporation — 53 с.
2. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains / Eric M. Hutchins , Michael J. Cloppert, Rohan M. Amin, Ph.D., — Lockheed Martin Corporation — 16 с.
3. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., & Stiller, B. (2010). An overview of IP flow-based intrusion detection. IEEE communications surveys & tutorials, 12(3), 343-356.
4. Графічна бібліотека ―graph-tool‖ [Електронний ресурс] — Режим доступу: https://graph-tool.skewed.de
5. (Bro/Zeek ATT&CK-based Analytics and Reporting) [Електронний ресурс]– Режим доступу: https://github.com/mitreattack/car/tree/master/implementations/bzar
6. M-Trends: the advanced persistent threat [Електронний ресурс] — Режим доступу : http://www.christiandve.com/wpcontent/uploads/2018/05/M-Trends.pdf
7. MITRE Cyber Analytics Repository [Електронний ресурс] — Режим доступу: https://car.mitre.org/
8. Опис файлової системи ―proc‖ [Електронний ресурс] — Режим доступу: http://man7.org/linux/man-pages/man5/proc.5.html
9. Saxe, J., & Sanders, H. (2014). Malware data science: attack detection and attribution. No Starch Press.
10. Тунелювання трафіка через DNS [Електронний ресурс] — Режим доступу: http://www.spy-soft.net/dns-tunneling/
11. ATT&CK Enterprise Matrix [Електронний ресурс] — Режим доступу: https:// https://attack.mitre.org/matrices/enterprise/

12. The Elements of Statistical Learning: Data Mining, Inference, and Prediction / Editors: D. Michie, D.J. Spiegelhalter, C.C. Taylor —February 17, 2016 — 298 c.
13. Aloqaily, M., Otoum, S., Al Ridhawi, I., & Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities. Ad Hoc Networks, 90, 101842.
14. Saxe, J., & Sanders, H. (2014). Malware data science: attack detection and attribution. No Starch Press.
15. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., & Stiller, B. (2010). An overview of IP flow-based intrusion detection. IEEE communications surveys & tutorials, 12(3), 343-356.
16. Alazab, M., Hobbs, S., Abawajy, J., & Alazab, M. (2012). Using feature selection for intrusion detection system. In 2012 International Symposium on Communications and Information Technologies (ISCIT) (pp. 296-301). IEEE.
17. Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective, 25(1-3), 18-31.
18. Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems, 100, 779-796.
19. Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.
20. ISO/IEC 27001:2013 Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. - Видання офіційне. - Женева: ISO, 2013. - 23 с.
21. Кузьмін, Д. Л. (2015). Забезпечення кібербезпеки критичної інфраструктури в умовах гібридних загроз. Бізнес Інформ, (3), 84-89.

22. Duque Anton, S., Fraunholz, D., Zemtsov, D., & Schotten, H. D. (2019). Hide and seek: Predicting cyber attacks with false alarms. In 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (pp. 439-448). IEEE.
23. Karami, M., Park, Y., & McCoy, D. (2016). Stress testing the booters: Understanding and undermining the business of DDoS services. In Proceedings of the 25th International Conference on World Wide Web (pp. 1033-1043).
24. Yan, J., Xu, G., Sun, Y., Qi, Y., & Chen, Q. (2018). Network-based intrusion detection system for cyber physical systems against false data injection attacks. Journal of Control Science and Engineering, 2018.
25. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.
26. Графічна бібліотека — graph-tool [Електронний ресурс] — Режим доступу: https://graph-tool.skewed.de

ДОДАТКИ

Додаток А

Оброблена та відсортована діяльність системних користувачів

9578 [eventfd] 1940
9578 /home/j3 970
9578 /lib/x86_64-linux-gnu/ld-2.23.so 970
9578 /lib/x86_64-linux-gnu/libc-2.23.so 970
9578 /lib/x86 64-linux-gnu/libdbus-1.so.3.14.6 970
9578 /lib/x86_64-linux-gnu/libdl-2.23.so 970
9578 /lib/x86_64-linux-gnu/libgcc_s.so.1 970
9578 /lib/ x86_64-linux-gnu/libgcrypt.so.20.0.5 970
9578 /lib/x86_64-linux-gnu/libglib-2.0.so.0.4800.2 970
9578 /lib/x86_64-linux-gnu/libgpg-error.so.0.17.0 970
9578 /lib/x86_64-linux-gnu/libjson-c.so.2.0.0 970
9578 /lib/x86_64-linux-gnu/liblzma.so.5.0.0 970
9578 /lib/x86_64-linux-gnu/libm-2.23.so 970
9578 /lib/x86_64-linux-gnu/libnsl-2.23.so 970
9578 /lib/x86_64-linux-gnu/libpcre.so.3.13.2 970
9578 /lib/x86 64-linux-gnu/libpthread-2.23.so 970
9578 /lib/x86 64-linux-gnu/libresolv-2.23.so 970
9578 /lib/x86_64-linux-gnu/librt-2.23.so 970
9578 /lib/x86_64-linux-gnu/libselinux.so.1 970
9578 /lib/x86_64-linux-gnu/libsystemd.so.0.14.0 970
9578 /lib/x86_64-linux-gnu/libwrap.so. 0.7.6 970
9578 pipe 5820
9578 /run/user/1000/speech-dispatcher/log/espeak.log 1940
9578 / run/user/ 1000/speech-dispatcher/log/speech-dispatcher.log 970
9578 / run/user/ 1000/speech-dispatcher/pid/speech-dispatcher.pid_ (deleted) 970
9578 type=STREAM 970
9578 /usr/lib/locale/C.UTF-8/LC_CTYPE 970
9578 /usr/lib/locale/locale-archive 970
9578 /usr/lib/speech-dispatcher-modules/sd _espeak 970
9578 /usr/lib/x86 64-linux-gnu/gconv/gconv-modules.cache 970
9578 /usr/lib/x86_64-linux-gnu/libasound.so.2.0.0 970
9578 /usr/lib/x86_64-linux-gnu/libasyncns.so.0.3.1 970
9578 /usr/lib/x86 64-linux-gnu/libdotconf.so.0.0.1 970
9578 /usr/lib/x86 64-linux-gnu/libespeak.so.1.1.48 970
9578 /usr/lib/x86_64-linux-gnu/libFLAC. so.8.3.0 970
9578 /usr/lib/x86_64-linux-gnu/libgthread-2.0.so.0.4800.2 970
9578 /usr/lib/x86_64-linux-gnu/libjack.so.0.1.0 970
9578 /usr/lib/x86 64-linux-gnu/libltdl.so.7.3.1 970
9578 /usr/lib/x86_64-linux-gnu/libogg. so.0.8.2 970
9578 /usr/lib/x86 64-linux-gnu/libportaudio. so.2.0.0 970
9578 /usr/lib/x86_64-linux-gnu/libpulse-simple.so. 0.1.0 970
9578 /usr/lib/x86 64-linux-gnu/libpulse. so.0.19.0 970
9578 /usr/lib/x86_64-linux-gnu/libsndfile.so. 1.0.25 970
9578 /usr/lib/x86 64-linux-gnu/libsonic. so.0.2.0 970
9578 /usr/lib/x86_64-linux-gnu/libstdc++.so. 6.0.21 970
9578 /usr/lib/x86_64-linux-gnu/libvorbisenc.so.2.0.11 970
9578 /usr/lib/x86 64-linux-gnu/libvorbis.so.0.4.8 970

9578 /usr/lib/x86 64-linux-gnu/libXau.so.6.0.0 970
9578 /usr/lib/x86_64-linux-gnu/libxcb.so. 1.1.0 970
9578 /usr/lib/x86_64-linux-gnu/libXdmcp.so. 6.0.0 970
9578 /usr/lib/x86_64-linux-gnu/pulseaudio/libpulsecommon-8.0.so 970
9578 /usr/lib/x86 64-linux-gnu/speech-dispatcher/spd pulse.so 970
9583 / 582
9583 / dev/shm/pulse-shm-2076972276 582
9583 / dev/shm/pulse-shm-3794718478 582
9583 [eventfd] 1164
9583 /home/j3 582
9583 /lib/x86_64-linux-gnu/ld-2.23.so 582
9583 /lib/x86_64-linux-gnu/libc-2.23.so 582
9583 /lib/x86 64-linux-gnu/libdbus-1.so.3.14. 6 582
9583 /lib/x86 64-linux-gnu/libdl-2.23.so 582
9583 /lib/x86 64-linux-gnu/libgcrypt.so.20.0.5 582
9583 /lib/x86_64-linux-gnu/libglib-2.0.so.0.4800.2 582
9583 /lib/x86 64-linux-gnu/libgpg-error.so.0.17.0 582
9583 /lib/x86_64-linux-gnu/libjson-c.so.2.0.0 582
9583 /lib/x86_64-linux-gnu/liblzma.so.5.0.0 582
9583 /lib/x86_64-linux-gnu/libm-2.23.so 582
9583 /lib/x86_64-linux-gnu/libsystemd.so.0.14.0 582
9583 /lib/x86_64-linux-gnu/libwrap.so. 0.7.6 582
9583 pipe 4656
9583 /run/user/1000/speech-dispatcher/log/dummy.log 1164
9583 /run/user/1000/speech-dispatcher/log/espeak.log 582
9583 / run/user/ 1000/speech-dispatcher/log/speech-dispatcher.log 582
9583 / run/user/ 1000/speech-dispatcher/pid/speech-dispatcher.pid_ (deleted) 582
9583 type=STREAM 582
9583 /usr/lib/speech-dispatcher-modules/sd _dummy 582
9583 /usr/lib/x86 64-linux-gnu/gconv/gconv-modules.cache 582
9583 /usr/lib/x86_64-linux-gnu/libFLAC. so.8.3.0 582
9583 /usr/lib/x86_64-linux-gnu/libgthread-2.0.so.0.4800.2 582
9583 /usr/lib/x86 64-linux-gnu/libltdl.so.7.3.1 582
9583 /usr/lib/x86_64-linux-gnu/libogg. so.0.8.2 582
9583 /usr/lib/x86_64-linux-gnu/libpulse-simple.so. 0.1.0 582
9583 /usr/lib/x86 64-linux-gnu/libpulse. so.0.19.0 582
9583 /usr/lib/x86_64-linux-gnu/libsndfile.so. 1.0.25 582
9583 /usr/lib/x86_64-linux-gnu/libvorbisenc.so.2.0.11 582
9583 /usr/lib/x86 64-linux-gnu/libvorbis.so.0.4.8 582
9583 /usr/lib/x86 64-linux-gnu/libXau.so.6.0.0 582
9583 /usr/lib/x86_64-linux-gnu/libxcb.so.1.1.0 582
9583 /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0 582
9583 /usr/lib/x86_64-linux-gnu/pulseaudio/libpulsecommon-8.0.so 582
9583 /usr/lib/x86 64-linux-gnu/speech-dispatcher/spd pulse.so 582
9583 / 582
9586 / dev/shm/pulse-shm-3225478376 582
9586 / dev/shm/pulse-shm-3794718478 582
9586 [eventfd] 1164
9586 /home/j3 582
9586 /lib/x86_64-linux-gnu/ld-2.23.so 582
9586 /lib/x86_64-linux-gnu/libc-2.23.so 582
9586 /lib/x86 64-linux-gnu/libdbus-1.so.3.14. 6 582

9586 /1ib/x86 64-1inux-gnu/libdl-2.23.so 582
9586 /1ib/x86 64-linux-gnu/libgcrypt.so.20.0.5 582
9586 /lib/x86_64-linux-gnu/libglib-2.0.so.0.4800.2 582
9586 /lib/x86 64-1inux-gnu/libgpg-error.so.0.17.0 582
9586 /1ib/x86_64-1inux-gnu/libjson-c.so.2.0.0 582
9586 /lib/x86_64-1inux-gnu/liblzma.so.5.0.0 582
9586 /lib/x86_64-1inux-gnu/1ibm-2.23.so 582
9586 /lib/x86_64-linux-gnu/libnsl-2.23.so 582
9586 /lib/x86_64-linux-gnu/libpcre.so.3.13.2 582
9586 /lib/x86 64-linux-gnu/libpthread-2.23.so 582
9586 /lib/x86 64-linux-gnu/libresolv-2.23.so 582
9586 /lib/x86_64-linux-gnu/librt-2.23.so 582
9586 /lib/x86_64-linux-gnu/libselinux.so.1 582
9586 /lib/x86_64-linux-gnu/libsystemd.so.0.14.0 582
9586 /lib/x86_64-linux-gnu/libwrap.so. 0.7.6 582
9586 pipe 6984
9586 /run/user/1000/speech-dispatcher/1og/cicero.log 1164
9586 /run/user/1000/speech-dispatcher/1og/dummy.log 582
9586 /run/user/1000/speech-dispatcher/1og/espeak.log 582
9586 / run/user/ 1000/speech-dispatcher/log/speech-dispatcher.log 582
9586 / run/user/ 1000/speech-dispatcher/pid/speech-dispatcher.pid_ (deleted) 582
9586 type=STREAM 582
9586 /usr/lib/speech-dispatcher-modules/sd _cicero 582
9586 /usr/lib/x86 64-linux-gnu/gconv/gconv-modules.cache 582
9586 /usr/lib/x86_64-linux-gnu/libasyncns.so.0.3.1 582
9586 /usr/lib/x86 64-linux-gnu/libdotconf.so.0.0.1 582
9586 /usr/lib/x86_64-linux-gnu/libFLAC. so.8.3.0 582
9586 /usr/lib/x86_64-linux-gnu/libgthread-2.0.so.0.4800.2 582
9586 /usr/lib/x86 64-linux-gnu/libltdl.so.7.3.1 582
9586 /usr/lib/x86_64-1inux-gnu/libogg. so.0.8.2 582
9586 /usr/lib/x86_64-linux-gnu/libpulse-simple.so. 0.1.0 582
9586 /usr/lib/x86 64-linux-gnu/libpulse. so.0.19.0 582
9586 /usr/lib/x86_64-linux-gnu/libsndfile.so. 1.0.25 582
9586 /usr/lib/x86_64-linux-gnu/libvorbisenc.so.2.0.11 582
9586 /usr/lib/x86 64-1inux-gnu/libvorbis.so.0.4.8 582
9586 /usr/lib/x86 64-linux-gnu/libXau.so.6.0.0 582
9586 /usr/lib/x86_64-linux-gnu/libxcb.so.1.1.0 582
9586 /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0 582
9586 /usr/lib/x86_64-linux-gnu/pulseaudio/libpulsecommon-8.0.so 582
9586 /usr/lib/x86 64-1inux-gnu/speech-dispatcher/spd pulse.so 582
9590 / 582
9590 / dev/shm/pulse-shm-1322794211 582
9590 / dev/shm/pulse-shm-3794718478 582
9590 [eventfd] 1164
9590 /home/j3 582
9590 /lib/x86_64-1inux-gnu/ld-2.23.so 582
9590 /lib/x86_64-linux-gnu/libc-2.23.so 582
9590 /1ib/x86 64-linux-gnu/libdbus-1.so.3.14. 6 582
9590 /1ib/x86 64-1inux-gnu/libdl-2.23.so 582
9590 /1ib/x86 64-linux-gnu/libgcrypt.so.20.0.5 582
9590 /lib/x86_64-linux-gnu/libglib-2.0.so.0.4800.2 582
9590 /lib/x86 64-1inux-gnu/libgpg-error.so.0.17.0 582

9590 /lib/x86_64-linux-gnu/libjson-c.so.2.0.0 582
9590 /lib/x86_64-linux-gnu/liblzma.so.5.0.0 582
9590 /lib/x86_64-linux-gnu/libm-2.23.so 582
9590 /lib/x86_64-linux-gnu/libnsl-2.23.so 582
9590 /lib/x86_64-linux-gnu/libpcre.so.3.13.2 582
9590 /lib/x86 64-linux-gnu/libpthread-2.23.so 582
9590 /lib/x86 64-linux-gnu/libresolv-2.23.so 582
9590 /lib/x86_64-linux-gnu/librt-2.23.so 582
9590 /lib/x86_64-linux-gnu/libselinux.so.1 582
9590 /lib/x86_64-linux-gnu/libsystemd.so.0.14.0 582
9590 /lib/x86_64-linux-gnu/libwrap.so. 0.7.6 582
9590 pipe 6984
9590 /run/user/1000/speech-dispatcher/log/cicero.log 582
9590 /run/user/1000/speech-dispatcher/log/dummy.log 582
9590 /run/user/1000/speech-dispatcher/log/espeak.log 582
9590 /run/user/1000/speech-dispatcher/log/generic.log 1164
9590 / run/user/ 1000/speech-dispatcher/log/speech-dispatcher.log 582
9590 / run/user/ 1000/speech-dispatcher/pid/speech-dispatcher.pid_ (deleted) 582
9590 type=STREAM 582
9590 /usr/lib/speech-dispatcher-modules/sd _generic 582
9590 /usr/lib/x86 64-linux-gnu/gconv/gconv-modules.cache 582
9590 /usr/lib/x86_64-linux-gnu/libasyncns.so.0.3.1 582
9590 /usr/lib/x86 64-linux-gnu/libdotconf.so.0.0.1 582
9590 /usr/lib/x86_64-linux-gnu/libFLAC. so.8.3.0 582
9590 /usr/lib/x86_64-linux-gnu/libgthread-2.0.so.0.4800.2 582
9590 /usr/lib/x86 64-linux-gnu/libltdl.so.7.3.1 582
9590 /usr/lib/x86_64-linux-gnu/libogg. so.0.8.2 582
9590 /usr/lib/x86_64-linux-gnu/libpulse-simple.so. 0.1.0 582
9590 /usr/lib/x86 64-linux-gnu/libpulse. so.0.19.0 582
9590 /usr/lib/x86_64-linux-gnu/libsndfile.so. 1.0.25 582
9590 /usr/lib/x86_64-linux-gnu/libvorbisenc.so.2.0.11 582
9590 /usr/lib/x86 64-linux-gnu/libvorbis.so.0.4.8 582
9590 /usr/lib/x86 64-linux-gnu/libXau.so.6.0.0 582
9590 /usr/lib/x86_64-linux-gnu/libxcb.so.1.1.0 582
9590 /usr/lib/x86_64-linux-gnu/libXdmcp.so.6.0.0 582
9590 /usr/lib/x86_64-linux-gnu/pulseaudio/libpulsecommon-8.0.so 582
9590 /usr/lib/x86 64-linux-gnu/speech-dispatcher/spd pulse.so 582
9593 / 776
9593 / dev/null 1164
9593 /lib/x86_64-linux-gnu/libc-2.23.so 388
9593 /lib/x86_64-linux-gnu/libc-2.23.so 388
9593 /lib/x86 64-linux-gnu/libdl-2.23.so 388
9593 /lib/x86_64-linux-gnu/libglib-2.0.so.0.4800.2 388
9593 /lib/x86_64-linux-gnu/libpcre.so.3.13.2 388
9593 /lib/x86 64-linux-gnu/libpthread-2.23.so 388
9593 pipe 3880
9593 /run/user/1000/speech-dispatcher/log/cicero.log 388
9593 /run/user/1000/speech-dispatcher/log/dummy.log 388
9593 /run/user/1000/speech-dispatcher/log/espeak.log 388
9593 /run/user/1000/speech-dispatcher/log/generic.log 388
9593 / run/user/ 1000/speech-dispatcher/log/speech-dispatcher.log 786
9593 / run/user/ 1000/speech-dispatcher/pid/speech-dispatcher.pid 388

9593 /run/user/ 1000/speech-dispatcher/pid/speech-dispatcher.pid_ (deleted) 388
9593 /run/user/ 1000/speech-dispatcher/speech.sock_ type=STREAM 388
9593 /usr/bin/speech-dispatcher 388
9593 /usr/lib/locale/locale-archive 388
9593 /usr/lib/x86 64-linux-gnu/gconv/gconv-modules.cache 388
9593 /usr/lib/x86 64-linux-gnu/libdotconf.so.0.0.1 388
9593 /usr/lib/x86 64-linux-gnu/libgmodule-2.0.so.0.4800.2 388
9593 /usr/lib/x86 64-linux-gnu/libpthread-2.0.so.0.4800.2 388
9597 / 96
9597 /proc/9597/exe 48
9639 / 388
9639 /proc/9639/exe 194
9640 / 388
9640 /proc/9640/exe 194
9654 / 344
9654 /proc/9654/exe 172
9655 / 388
9655 /proc/9655/exe 194
9656 / 388
9656 /proc/9656/exe 194
9657 / 214
9657 /proc/9657/exe 107
9658 / 122
9658 /proc/9658/exe 61
9661 / 234
9661 /proc/9661/exe 117
9763 / 388
9763 /proc/9763/exe 194
9768 / 388
9768 /proc/9768/exe 194
9910 / 62
9910 /proc/9910/ exe 31
9980 / 380
9980 /proc/9980/ exe 166
9981 / 388
9981 /proc/9981/exe 194
::999::Backgro.kSource 4179 195
::999::clientcursormon 4179 195
::999::Collect.xecutor 4179 195
::999::conn 66 4179 19
::999::conn 66 4179 27
::999::conn 66 4179 19
::999::conn 66 4179 11
::999::conn 66 4179 8
::999::DeadlineMonitor 4179 195
::999::FlowCon.fresher 4179 195
::999::FreeMonHTTP-0 4179 195
::999::FreeMonNet 4179 195
::999::FreeMon.ocessor 4179 195
::999::ftdc 4179 195
::999::listener 4179 195
::999::Logical.cheReap 4179 195

::999::Logical.Refresh 4179 195
::999::mongod 4179 1950
::999::Periodi.kRunner 4179 195
::999::signalP.gThread 4179 195
::999::startPe.actions 4179 195
::999::startPe.ressure 4179 195
::999::Timesta.Monitor 4179 195
::999::TTLMonitor 4179 195
::999::waitForMajority 4179 195
::999::WTCheck.tThread 4179 195
::999::WTIdleS.Sweeper 4179 195