НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: Метод аналізу APT атак за допомогою технік MITRE та алгоритму нечіткого пошуку

Виконала: студентка 2 курсу, групи ФБ-81мп
(шифр групи)

Кифоренко Ірина Вячеславівна _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н., доц. Барановський О. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (освітньо-професійна програма) – 125 Кібербезпека («Системи, технології та математичні методи кібербезпеки)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)

«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Кифоренко Ірині Вячеславівні

1. Тема дисертації: Метод аналізу APT атак за допомогою технік MITRE та алгоритму нечіткого пошуку
науковий керівник дисертації: доц. Барановський Олексій Миколайович, кандидат технічних наук, доцент
затверджені наказом по університету від «___» _____ 2019 р. № _____

2. Термін подання студентом дисертації 10.12.2019 р.

3. Об’єкт дослідження : події в операційній системі Windows, що були створені внаслідок реалізації APT атаки.

4. Вихідні дані : метод аналізу APT атак за допомогою технік MITRE та алгоритму нечіткого пошуку та його практична реалізація, побудована хронологія подій атаки, що отримана внаслідок застосування створеного методу до набору початкових даних.

5. Перелік завдань, які потрібно розробити : 1.Проаналізувати APT атаки та її етапи життєвого циклу. 2.Порівняти існуючі методи виявлення та аналізу APT атак та визначити їх основні недоліки. 3.Вирізнити із основного опису MITRE технік їхні можливі варіанти реалізації та на основі цього описати їхні індикатори. 4.Розробити механізм для кореляції подій за певними параметрами, отриманих на етапі розрізнення використання MITRE технік у розрізі атаки за допомогою алгоритму нечіткого пошуку. 5.Побудувати хронологію APT атаки на основі отриманих спрацювань за MITRE техніками.

6. Орієнтовний перелік ілюстративного матеріалу : 9 ілюстрацій, 42 таблиці.

7. Орієнтовний перелік публікацій _____
_____
_____

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз методів виявлення APT атак. Опис, порівняння у таблицях.		
2	Розгляд MITRE та алгоритмів нечіткого пошуку. Формування методу аналізу.		
3	Створення тестів для матриці MITRE.		
4	Визначення індикаторів технік та їх документування.		
5	Створення кореляційних правил в SIEM.		
6	Розробка механізму пошуку подій за алгоритмом Вагнера-Фішера.		
7	Реалізація парсингу подій та механізмів візуалізації хронології.		
8	Вивантаження спрацювань правил та застосування методу аналізу.		
9	Опис роботи методу та опис його програмної реалізації.		
10	Аналіз результатів методу, документування.		

Студент _____ І.В. Кифоренко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ О.М. Барановський
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація складається зі вступу, 5 розділів, висновків, переліку посилань та додатків. Обсяг роботи складає 129 сторінок, у роботі міститься 9 ілюстрацій, 42 таблиці, 22 джерела за переліком посилань.

Об’єктом дослідження є події в операційній системі Windows, що були створені внаслідок реалізації APT атаки.

Предметом дослідження є методи та підходи до аналізу APT атак.

Метою роботи є створення методу аналізу APT атаки для визначення її етапів та основних індикаторів компрометації за допомогою покращеного аналізу подій і кореляції подій між собою.

Методами дослідження було обрано аналіз етапів життєвого циклу APT атаки та їх особливостей, аналіз та порівняння методів та підходів до аналізу атак, дедуктивний метод для визначення індикаторів технік MITRE у явному вигляді та гіпотетико-дедуктивний метод під час порівняння теоретичних індикаторів із наявними у подіях операційної системи, експеримент для емуляції технік та етапів APT атаки задля отримання подій та їх подальшого аналізу.

Результатами роботи є створений метод аналізу APT атак за допомогою технік MITRE та алгоритму нечіткого пошуку.

Апробація результатів дисертації була висвітлена у статті журналу Фізико-Технічного Інституту «THEORETICAL AND APPLIED CYBERSECURITY» під назвою «Method of APT attack analysis using MITRE techniques and Wagner-Fischer algorithm».

Ключові слова: APT атака, Kill Chain, Diamond Model, MITRE, техніка, тактика, нечіткий пошук, хронологія подій.

ABSTRACT

The master's dissertation consists of an introduction, 5 sections, conclusions, a list of references and appendices. The volume of work is 129 pages, the work contains 9 figures, 42 tables, 22 sources in the list of references.

The object of the study is set of events in the Windows operating system that were created as a result of the APT attack.

The subject of the study consists of methods of the APT attacks analysis.

The purpose of the work is to create a method of APT attack analysis to determine its stages and main indicators of compromise through improved event analysis and event correlation.

The research methods were used are analyzing the APT attack lifecycle stages and their features, analyzing and comparing methods and approaches to attack analysis, using deductive method for defining MITRE technique indicators, using hypothetical-deductive method in comparing theoretical indicators with existing operating system events, conducting of an experiment to emulate APT attack techniques to capture events and analyze them further.

The result of dissertation is a method of APT attack analysis using MITRE techniques and fuzzy search algorithm.

The results of dissertation were covered in the article of the Institute of Physics and Technology Journal "THEORETICAL AND APPLIED CYBERSECURITY" entitled "Method of APT attack analysis using MITER techniques and Wagner-Fischer algorithm".

Key words: APT attack, Kill Chain, Diamond Model, MITRE, technique, tactic, fuzzy string search, event timeline.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

APT	Advanced Persistent Threat;
APA	Advanced Persistent Adversary;
Sysmon	System Monitor By Mark Russinovich and Thomas Garnier;
IoC	Indicator of Compromise;
SIEM	Security Information and Event Management;
DLL	Dynamic Link Library;
C&C	Command and Control;
APA	Advanced Persistent Adversary.

ВСТУП

У міру стрімкого розвитку інформаційних технологій та їх поширення у різних сферах діяльності приватних та державних організацій зростає кількість загроз та проблем у сфері інформаційної безпеки. У минулому хакери атакували інформаційні системи, більше керуючись особистими та фінансовими мотивами. Однак сьогодні існують інші мотиви, такі як політичні, і вони потенційно підтримуються урядами чи певними організаціями для розпалення інформаційних війн та маніпулювання даними у корисливих цілях. Одним із поширених засобів для таких цілей є APT атаки. APT використовують різні складні методи та прийоми, що застосовуються до певної цільової організації, щоб викрасти конфіденційну та іншу чутливу інформацію. Ряд антивірусних корпорацій та організацій із інформаційної безпеки докладають зусиль для аналізу таких атак, як APT, щоб створити засоби запобігання та захисту від атак APT.

Питання аналізу APT насьогодні постає досить гострим питанням, так як існує лише невелика кількість методів для їх аналізу, для яких достатньо проблематично розробити підхід для практичного застосування. Тому створення методу аналізу даних атак є цінним та важливим завданням, що може полегшити роботу для багатьох аналітиків з інформаційної безпеки та розкрити нові можливості для подальшого розвитку методів раннього виявлення APT та розробки засобів захисту.

Об’єктом дослідження є події в операційній системі Windows, що були створені внаслідок реалізації APT атаки.

Предметом дослідження є методи та підходи до аналізу APT атак.

Метою роботи є створення методу аналізу APT атаки для визначення її етапів та основних індикаторів компрометації за допомогою покращеного аналізу подій і кореляції подій між собою.

Для досягнення цієї мети були поставлені такі завдання:

1. Проаналізувати APT атаки та її етапи життєвого циклу.

2. Порівняти існуючі методи виявлення та аналізу APT атак та визначити їх основні недоліки.
3. Вирізнити із основного опису MITRE технік їхні можливі варіанти реалізації та на основі цього описати їхні індикатори.
4. Розробити механізм для кореляції подій за певними параметрами, отриманих на етапі розрізнення використання MITRE технік у розрізі атаки за допомогою алгоритму нечіткого пошуку.
5. Побудувати хронологію APT атаки на основі отриманих спрацювань за MITRE техніками.

Методами дослідження було обрано аналіз етапів життєвого циклу APT атаки та їх особливостей, аналіз та порівняння методів та підходів до аналізу атак, дедуктивний метод для визначення індикаторів технік MITRE у явному вигляді та гіпотетико-дедуктивний метод під час порівняння теоретичних індикаторів із наявними у подіях операційної системи, експеримент для емуляції технік MITRE та етапів APT атаки задля отримання подій та їх подальшого аналізу.

Наукова новизна дослідження полягає у тому, що був запропонований метод і його практична реалізація для аналізу APT атаки на конкретних інформаційних подіях в операційній системі Windows із візуалізацією взаємозв’язків послідовності етапів APT атаки. Було створено новий метод аналізу, запропоновано новий підхід до встановлення взаємозв’язків між подіями, які були виявлені шляхом фільтрації із використанням технік MITRE.

Практичне значення одержаних результатів полягає у застосуванні індикаторів технік MITRE як правил для спрацювання у SIEM Security ArcSight ESM від Micro Focus та методу у розрізі аналізу потенційних APT атак для побудови хронології подій згідно із матрицею MITRE.

Апробація результатів дисертації була висвітлена у статті журналу НТУУ КПІ ім.. Ігоря Сікорського, Фізико-Технічного Інституту «THEORETICAL AND APPLIED CYBERSECURITY» під назвою «Method of APT attack analysis using MITRE techniques and Wagner-Fischer algorithm».

1 МЕТОДИ ВИЯВЛЕННЯ APT АТАК ТА ЇХ ПРОБЛЕМАТИКА

У першому розділі розглянуто методи аналізу APT атак, проведено аналіз методів, створено їх порівняльну характеристику та визначено основну проблематику сучасних методів аналізу. На основі визначених недоліків та не вирішених проблем існуючих методів було обрано основні параметри, які мають бути присутні у розробленому методі аналізу APT атак.

1.1 Методи виявлення за моделями Kill Chain, Diamond

APT – термін, який застосовують для опису атак, в яких зловмисники або їх група встановлює не легітимне та довгострокове втручання в мережу жертви задля отримання конфіденційної інформації або з іншими цілями.

Згідно з [1, 2] APT атака поєднує три основні складові, за якими стоїть певне значення у розрізі атаки: «Advanced» – використовується широкий спектр різних методів та інструментів, «Persistent» – підтримка розширеної присутності у внутрішньому середовищі цілі, щоб постійно отримувати та вилучати потенційно цінні дані із середовища, «Threat» – APT атакує організації для досягнення певної цілі і часто зосереджується на отриманні і вилученні фінансової, технологічної та іншої інформації, надаючи своїм спонсорам конкурентну перевагу.

За визначенням, даним Національним інститутом стандартів і технологій США (NIST), APT описується у комплексному вигляді [3]: «Противник має високий рівень знань та значні ресурси, які дозволяють йому створити можливості для досягнення своїх цілей за допомогою декількох векторів нападу. Ці цілі, як правило, включають встановлення та розширення присутності у інфраструктурі цільових організацій з метою вилучення інформації, підриву або перешкоджання критичним аспектам цільової організації. Попередня стійка загроза переслідує свої цілі неодноразово протягом тривалого періоду,

пристосовується до зусиль захисників протистояти цьому та має намір підтримувати рівень взаємодії, необхідний для задоволення своїх цілей".

У Таблиці 1.1 було зведено порівняння APT атак із традиційними атаками.

Таблиця 1.1 – Порівняння APT атак із традиційними атаками

	Традиційна атака	APT атака
Порушник	Зазвичай одна або декілька людей	Визначена високоорганізована група людей, яка забезпечена потрібними ресурсами для проведення атаки
Ціль	Невизначена, переважно індивідуальна система	Конкретні організації, урядові установи, комерційні підприємства
Мета	Фінансові вигоди, демонстрація здібностей	Фінансові вигоди, конкурентні та стратегічні переваги
Підхід	Короткострокова, одноразова	Неодноразові спроби, пристосовуються до опору захисним засобам, тривалий термін

Під час пошуку наявних методів аналізу APT було виявлено такі моделі: Kill Chain та Diamond, які є одними із основних та найбільш популярних моделей для виявлення та аналізу вторгнень.

Модель Kill Chain – це один із методів виявлення вторгнень. Вона містить 7 фаз, які дають змогу помітити саме вторгнення та дозволяють краще зрозуміти тактику, прийоми та процедури противника. Фази утворюють взаємопов’язаний кінцевий процес, який описаний як ланцюг на Рис.1.1, у якому порушення будь-якого кроку перериває весь ланцюжок. Типовий APT проходить через сім етапів (фаз): розвідка (Recon), озброєння (Weaponization), доставка (Delivery), експлуатація (Exploitation), інсталяція (Installation), командування і контроль (Command and Control), дії щодо цілей (Exfiltration).

Рисунок 1.1 – Модель Kill Chain

Модель визначає, що повинні зробити противники для досягнення своєї мети шляхом проникнення в мережу, вилучення даних та підтримки своєї присутності в організації. Завдяки цій моделі стало зрозуміло, що при зупинці противника на будь-якій із стадій дає змогу зупинити весь процес атаки. Для досягнення успіху супротивники мають повністю від початку до кінця пройти усі фази моделі. Натомість фахівці з інформаційної безпеки для досягнення успіху зі своєї сторони мають заблокувати діяльність зловмисників на будь-якій стадії, але більш успішним буде припинення атаки на більш ранніх фазах.

Модель Kill Chain містить 7 етапів, котрі мають свої особливості [7]:

1. Перший етап – розвідка. На цьому етапі зловмисник оцінює ціль за межами організації з технічної та не технічної точки зору. Зловмисник працює над тим, щоб визначити, які цільові організації зможуть принести найбільшу користь за ті ресурси, які будуть вилучені із мереж цільової жертви. Зловмисник буде шукати інформаційні системи з невеликим захистом або експлуатованими вразливими місцями.

Активний збір інформації: за допомогою активного збору інформації та розвідки зловмисник активно взаємодіє з цільовою системою. Прикладом активного збору інформації та розвідки є сканування портів, де зловмисник працює над перерахуванням наявних відкритих портів у мережі. У такому випадку мета зловмисника полягає в тому, щоб виявити порти, вразливі до експлуатації, і, таким чином, забезпечити доступ до цільової системи.

Збір пасивної інформації: пасивний збір інформації стосується цільових інформаційних систем, без залучення цих інформаційних систем безпосередньо.

2. Другий етап ланцюга – це озброєння. Під час озброєння противник розробляє зловмисне програмне забезпечення, яке спеціально створене для вразливості, виявленої під час фази розвідки цільової системи. На основі інформації, зібраної на етапі розвідки, зловмисник налаштовує набір інструментів, щоб відповідати конкретним вимогам цільової мережі.

3. Третя стадія в Kill Chain, доставка, включає передачу коду APT від зловмисника до цільової інформаційної системи для експлуатації. На основі сучасних досліджень та аналізу звітного розслідування Verizon 2018 року, найбільш популярним способом доставки є фішинг, спрямований на внутрішнього співробітника організації.

Ретельно досліджена та розроблена кампанія фішингу проти організації на основі інформації, зібраної на етапі розвідки, призведе до того, що працівники організації виконають код зловмисного програмного забезпечення APT у своїх інформаційних системах. Фішинг повідомлення, швидше за все, міститиме вкладення, таке як Microsoft Word або документ Adobe PDF. Вкладення міститиме код, який при виконанні призведе до того, що APT закріпиться в організаційній мережі.

4. Під час фази експлуатації зловмисне програмне забезпечення APT виконується в цільовій мережі за допомогою віддалених або локальних механізмів, використовуючи переваги виявлених вразливостей для отримання доступу суперкористувача до цільової інформаційної системи.

5. Наступний етап це інсталяція. Після успішної експлуатації системи код зловмисного програмного забезпечення APT встановиться на цільову інформаційну систему. Після цього шкідливі програми APT почнуть завантажувати додаткове програмне забезпечення, якщо доступ до мережі є. Це дозволяє завантаженому програмному забезпеченню залишатись невеликим та непомітним.

6. Командування та управління, також ця стадія відома як C2, - це коли зловмисник встановив свій APT код управління та зв'язку на цільову мережу. Це програмне забезпечення дозволяє зловмиснику повною мірою керувати кодом APT у цільовому середовищі та дозволяє йому просуватися глибше в мережу, розшифровувати дані та проводити знищення чи вилучення даних або відмову в сервісних операціях.

7. Дії щодо цілей. Дії та завдання APT залежать від його конкретної місії. APT може бути зосереджений на знищенні даних, відмові в обслуговуванні або вилученні чутливих даних.

Що стосується ексфільтрації даних, APT атака може бути зацікавлена у власних організаційних даних, таких як інженерні проекти або особисті відомості працівника та замовника. У разі відмови в обслуговуванні, як і відключення електроенергії в Україні, грудень 2015 року, APT може відключити ключовий компонент інфраструктури організації тимчасово порушити послуги.

Наступною розглянутою моделлю є Diamond модель, яка описує як порушник застосовує можливості в певній інфраструктурі проти своєї жертви. Такі заходи називаються подіями і є атомарними елементами в зазначеній моделі аналізу вторгнень. Подія – це будь-яка зафіксована зміна стану системи.

Однією із переваг цієї моделі є те, що вона забезпечує ефективний (але не обов'язково комплексний) список характеристик, які повинні бути присутніми у кожній події. Тому, після документування події з усією наявною інформацією, будь-які порожні характеристики мають бути заповнені шляхом аналізу додаткових характеристик.

Кожну подію можна описати за допомогою чотирьох характеристик [6]: порушник (адреси електронної пошти, телефонні номери; мережеві активи); інфраструктура (IP-адреси, доменні імена, адреси електронної пошти); жертва (особи, активи мережі, адреси електронної пошти); можливості (шкідливе програмне забезпечення та інший інструментарій).

Також можна виділити такі мета-характеристики, які розширюють відомості про подію: час (початок та завершення); фаза; результат

(успішність/неуспішність); напрям; методологія (клас діяльності); ресурси (необхідні дані/засоби для проведення заходу).

Мета-характеристики дозволяють упорядкувати події у каналі активності, групувати схожі події різним чином та виділяти критичні дані, де можливо.

Також можна виділити таке поняття як довірче значення. Будь-яка характеристика події має довірче значення. Дане значення заздалегідь залишається невизначеним, так як кожна модель впровадження може визначати довірчий рівень по-різному – її можна виразити як функцію декількох значень (впевненість в аналітичному висновку, точність джерела даних).

E = [(Порушник, Довірче значення $_{порушника}$), (Можливість, Довірче значення $_{можливості}$), (Інфраструктура, Довірче значення $_{інфраструктури}$), (Жертва, Довірче значення $_{жертви}$), (Час $_{початок}$, Довірче значення $_{час\ початку}$), (Час $_{кінець}$, Довірче значення $_{час\ кінця}$), (Фаза, Довірче значення $_{фази}$), (Результат, Довірче значення $_{результату}$), (Напрям, Довірче значення $_{напряму}$), (Методологія, Довірче значення $_{методології}$), (Ресурси, Довірче значення $_{ресурсів}$)].

Подія E формально визначається n-кортеж, де кожен елемент кортежу є значенням характеристики в поєднанні із незалежним довірчим значенням.

На Рис.1.2 зображено дану модель аналізу вторгнень. Модель складається із основних характеристик події вторгнення: порушник, можливість, інфраструктура та жертва. Основні характеристики пов'язані за допомогою ліній, щоб відобразити фундаментальні зв'язки між характеристиками, які можуть бути використані аналітично для подальшого виявлення та аналізу зловмисної діяльності.

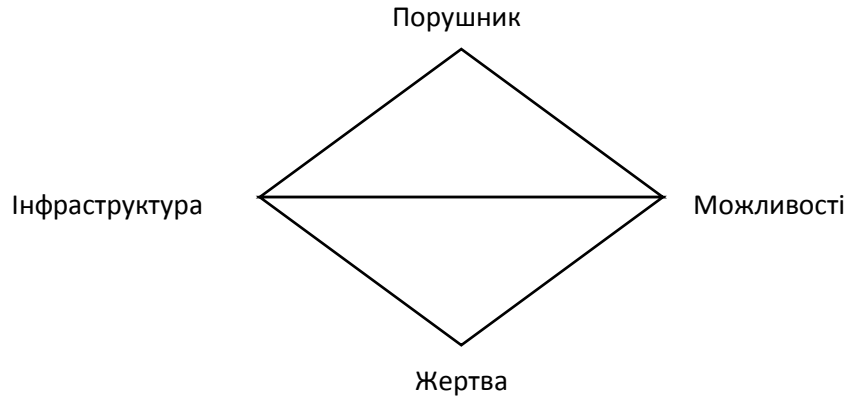

Рисунок 1.2 – The Diamond Model – модель аналізу вторгнень

У будь-якому випадку порушник не здатний скоїти усі заплановані зловмисні дії проти жертви у межах однієї події, тому його зловмисна діяльність описується низкою подій у межах набору упорядкованих фаз, в яких, як правило, кожна фаза повинна бути успішно виконана для досягнення свого наміру. Потік активності – це направлений фазово-упорядкований граф, де кожна вершина є подією, а дуги (тобто спрямовані ребра) ідентифікують причинні зв'язки між подіями. Дуги позначаються аналітичним довірчим значенням, встановлюючи причинно-наслідкові зв'язки, чи є шлях ТА (необхідний) або АБО (необов'язковий – існує декілька потенційних шляхів від події), чи є дуга актуальним станом або гіпотезою.

Потоки організовані вертикально таким чином, що кожен потік описує всі причинні події, які порушник виконує проти певної жертви (однак реалізація моделі визначає функцію жертви), колективно спрямовану на виконання наміру порушника. Тому кожен потік є специфічним для однієї пари порушник-жертва, хоча в багатьох випадках потоки активності можуть дещо відрізнятися для інших жертв, оскільки супротивник об'єднує інфраструктуру, процеси та можливості для зменшення витрат.

1.2 Порівняння методів

Під час аналізу атаки за допомогою обох моделей використовується принцип розділення атаки за життєвим циклом APT.

Залежні події (що складаються з жертви, супротивника, можливості, інфраструктури) створюють потоки активності у моделі Kill Chain. Ці потоки складають (за допомогою аналізу ключових показників кампанії) змагальні кампанії. Перший взаємозв'язок між двома моделями.

Аналіз Diamond Model та Kill Chain сильно доповнюють один одного. Аналіз Kill Chain дозволяє аналітику "націлити та залучити супротивника для створення бажаних ефектів". Diamond дозволяє аналітикам розвивати

торговельний апарат і розуміння для побудови та організації знань, необхідних для виконання аналізу ланцюга атаки.

Як тільки аналітик розробляє потік активності, курси дій для кожної події вздовж потоку можуть бути визначені за допомогою матриці курсу Kill Chain. Курси дій для кожного з етапів Kill Chain визначаються для потоків активності. Сила Diamond Model полягає в тому, що курси дій можуть бути розроблені, щоб охопити кілька жертв і всю діяльність противника, що зробить дії ще більш потужними, оскільки вони ще більше знижують спроможність противника.

Групи активності, згруповані тим самим ймовірним супротивником (тобто кластеризація за параметрами) з аналізом найбільшої загальної функції, встановленої серед подій у групі, можуть забезпечити необхідні ключові показники кампанії Kill Chain, необхідні для фокусування та визначення пріоритетних напрямків дій.

При роботі з Cyber Kill Chain над окремими подіями існує ризик припустити, що відсутність будь-якої подальшої зловмисної активності означає, що діючі захисні заходи спрацювали. Однак, використовуючи Diamond Model, є можливість зв'язати подія з іншими і визначити потенційні наступні дії.

Тому застосування цих двох моделей окремо може бути недостатньо ефективним у процесі виявлення слідів APT атаки у подіях та при подальшому аналізі цих подій. Зважаючи на це, багато експертів наполягають на використанні цих моделей у комплексі для забезпечення більш якісного аналізу атак та ефективного пошуку вразливостей та слабких місць інфраструктури жертви, які були експлуатовані в атаці.

Під час аналізу методів було знайдено ряд їх переваг та недоліків, які впливають на хід аналізу APT атак та можуть одночасно як перешкоджати аналізу, так і сприяти йому. Визначення плюсів та мінусів моделей дозволяють краще зрозуміти, чому той чи інший метод можна використовувати або треба вдосконалювати.

Для порівняння наявних методів визначені їх переваги та недоліки описано у Таблиці 1.2.

Таблиця 1.2 – Переваги та недоліки проаналізованих методів

Метод	Переваги	Недоліки
Kill Chain	1.Використовується основними авторитетними організаціями під час аналізу. 2.Ідентифікація точок вторгнення для подальшого розривання ланцюгу атаки. 3.Графічне відображення ланцюгу подій є чітким та зрозумілим. 4.Присутнє встановлення хронології подій. 5.Присутнє визначення зловмисника, його дій та цілей.	1.Відсутній чіткий підхід до пошуку зловмисних подій. 2.Немає схеми пов’язування подій. 3.Присутній ризик не виявлення потенційно небезпечної діяльності. 4.Бажано використовувати у поєднанні із іншою моделлю.
Diamond Model	1.Простота моделі – містить всього 4 ключові параметри. 2.Огляд та розуміння усіх взаємодій у зловмисника для розуміння природи загрози. 3.Можливість пов’язувати події і визначати наступні потенційні дії. 4.Присутнє встановлення хронології подій атаки. 5.Присутнє визначення зловмисника, його дій та цілей за ключовими параметрами моделі.	1.Без попереднього детального ознайомлення із моделлю важко будувати ланцюг подій. 2.Відсутній чіткий підхід до пошуку зловмисних подій. 3.Присутній ризик не виявлення потенційно небезпечної діяльності. 4.Рекомендовано застосовувати у поєднанні із іншою моделлю для розкриття сутності усієї атаки з різних сторін.

Також для порівняння методів аналізу було обрано набір параметрів, за якими здійснювався порівняльний аналіз на основі отриманих переваг та

недоліків. Цими 8 параметрами було визначено: побудова хронології подій, встановлення зв’язків між подіями, наявна схема пошуку зловмисних дій, співставлення подій із життєвим циклом APT, легкість в імплементації методу при аналізі, наявність готової програмної реалізації методу у чистому вигляді методу без доповнень, визначення слабких/вражених місць жертви, визначення зловмисника, профіля жертви, задіяних інструментів.

З огляду на обрані характеристики або ж параметри для зрівняння методів аналізу було проведено аналітику обраних методів та визначено присутність чи відсутність параметрів для кожного із методів. За умовними позначеннями у таблиці: «+» – якщо даний параметр присутній у методі, «-» – якщо даний параметр відсутній у методі. Порівняльна характеристика за параметрами подана у Таблиці 1.3.

Таблиця 1.3 – Порівняння методів за переліком параметрів

№	Параметр порівняння	Kill Chain	Diamond Model
1	Побудова хронології подій	+	+
2	Встановлення зв’язків між подіями	-	+
3	Наявна схема пошуку зловмисних дій	-	-
4	Співставлення подій із життєвим циклом APT	+	+
5	Легкість в імплементації методу при аналізі	-	-
6	Наявність готової програмної реалізації методу у чистому вигляді методу без доповнень	-	-
7	Визначення слабких/вражених місць жертви	+	+
8	Визначення зловмисника, профіля жертви, задіяних інструментів	+	+

Як бачимо, дані методи аналізу не є досконалими і потребують певних змін, також виявлено, що навіть при використанні даних методів рекомендовано

застосовувати їх у поєднанні для підвищення їх ефективності та зменшення ризиків втрати важливих ключових моментів атаки. Не існує жодного методу, який покриває усі вказані параметри, що є важливою проблемою у сфері розслідування кібер злочинів для відновлення ланцюгів зловмисних дій у хронології подій APT атак. Наявність даної проблеми створює потребу у розробці нового методу для аналізу APT, який можна буде прикладним чином застосовувати для розслідування злочинів у кібер просторі.

На основі даних із таблиці вище, можна зробити висновок, що в цілому проаналізовані методи мають такі основні недоліки: відсутність схеми пошуку подій, складність в імплементації методів при аналізі, відсутність готової програмної реалізації для часткової або ж повної автоматизації аналізу APT.

З точки зору аналітика, найбільшим слабким місцем методів є відсутність методики пошуку подій, про що буде йти мова у наступному підрозділі.

1.3 Проблеми існуючих методів

Коли справа підходить до детального аналізу подій, аналітик стикається із великою проблемою – як серед тисяч подій знайти саме ті, які свідчать про проведення тієї чи іншої атаки. Для цього і потрібна модель виявлення вторгнень. Перш за все, за допомогою цієї моделі можна уявно розділити атаку на певну кількість стадій, які мають свої особливості та методи, що застосовуються для реалізації кожного із кроків. Kill Chain містить тільки фактичний опис фаз атаки, але не виділяє чіткі об’єкти та суб’єкти атаки, натомість Diamond модель оперує такими поняттями як порушник, можливість, інфраструктура та жертва, де на основі цього визначаються можливі етапи атаки та застосовується підхід виділення цих етапів, як і в Kill Chain. Етапи визначені та зрозумілі, але відсутні певні маркери, що можуть показати, де саме в подіях операційної системи знаходяться ці фази. У Kill Chain та Diamond моделях відсутня конкретика у

методах та техніках, що застосовують противники на стадіях моделей, тобто конкретний метод пошуку відсутній.

Тобто існують моделі для визначення потенційних узагальнених стадій APT атаки і під них подається опис дій, що можуть вчинити зловмисники на цих етапах. Такий відкритий ресурс, як MITRE ATT&CK, дає більш детальний опис цих дій у вигляді матриці із тактиками (етапи атаки) та техніками у розрізі кожної із тактик. Але їхній опис також достатньо узагальнений, тому що варіантів застосування однієї техніки безліч. В описі MITRE пропонуються ресурси, де можуть бути знайдені потрібні події, але вони також подаються у загальному вигляді, наприклад, Моніторинг API, моніторинг DLL, моніторинг процесів, реєстр Windows, журнали подій Windows. Але у журналі подій Windows є зокрема 3 основних розділи: Application, Security та System. І кожен із розділів містить різний розподіл подій за ідентифікаторами. Розглянуті моделі у кінцевому результаті можуть спиратися і на техніки MITRE ATT&CK без вказування на ключові індикатори атаки, що були знайдені і яким чином.

MITRE ATT&CK – це відкрита база знань про тактики та техніки, які застосовуються для реалізації різних атак в інформаційному просторі [8]. База знань ATT&CK використовується як основа для розробки конкретних моделей загроз та методів їх виявлення чи аналізу у приватному секторі, в державних організаціях та у сфері послуг кібербезпеки.

MITRE ATT&CK містить набір технік, які описані у розрізі тактик – етапів життєвого циклу APT атак. Дана особливість матриці MITRE дозволяє розглядати атаки як послідовний набір використаних технік щодо цілі. Тому застосування технік як підходу до пошуку ключових моментів APT атак є ефективнішим способом, аніж пошук індикаторів «в лоб», а також дозволяє чітко розділяти етапи атаки і розділяти їх за життєвим циклом APT.

З огляду на присутність ряду ключових проблем в існуючих методах аналізу APT атак – відсутність алгоритму пошуку зловмисних подій та дій, висока складність у впровадженні методів для проведення аналізу, відсутність програмної реалізації методів, який здатен забезпечити часткову або ж повну

автоматизацію аналізу APT на основі певного набору даних – виникла потреба у розробці нового методу для аналізу APT атак та покриття усіх наявних недоліків існуючих методів або ж моделей.

На основі цього була поставлена мета розробити такий метод, що буде, перш за все, прикладним у застосуванні, із чітким описом потенційно небезпечних індикаторів технік зловмисників, підходом до пошуку зловмисних дій у заданому наборі даних, із автоматичною побудовою хронології подій в APT, можливістю відсіювання фальш-позитивних спрацювань, а також із найголовнішим та не менш важливим елементом – алгоритмом пошуку взаємозв’язків між техніками для побудови хронології подій всієї атаки та для графічного відображення різних взаємозв’язків для заданих параметрів події. Для реалізації цього також була поставлена першочергова задача пошуку найбільш ефективного алгоритму пов’язування подій та підходу до фільтрації даних.

Висновки до розділу 1

APT атака – це тип цільової атаки. Цільові атаки використовують найрізноманітніші методи, наприклад, такі як завантаження за допомогою драйверів, введення SQL ін’єкцій, застосування шкідливих програм, шпигунських програм, фішингу та спаму і т.д.. APT завжди вважається цільовою атакою, але цільова атака не обов'язково є APT атакою.

В цілому вирізняють такі етапи APT атаки:

1. Вибір цілі та збір інформації.
2. Атака та компрометація системи
3. Встановлення C&C каналу зв'язку
4. Авторизація та викрадення облікових даних
5. Захоплення та обробка даних
6. Забезпечення стійкості до засобів/методів запобігання атаки
7. Приховування слідів

У якості моделей аналізу вторгнень були розглянуті Kill Chain та Diamond. Основною суттю даних моделей є визначення чітких етапів проходження APT атаки та опису дій, які індивідуально були застосовані на кожному із етапів зловмисниками певної атаки. Diamond також пропонує брати за основу 4 основні сутності для опису атаки та її взаємодій, а також додаткових мета-характеристик, які допомагають більше конкретизувати даний опис. Саме порушник, можливість, інфраструктура та жертва є основними поняттями даної моделі, навкруги яких формується той чи інший етап APT.

Але, якщо не зважати на багатий теоретичний опис даних моделей та їх можливість розкривати сутність атак, було зроблено висновок, що вони мають декілька суттєвих недоліків, що впливають на результати застосування цих моделей з практичної точки зору. Насамперед, це відсутність чіткої практичної частини застосування моделі. Не розкривається підхід до самого аналізу, з чого його починати і що потенційно треба шукати, де саме і які події потрібно брати під розгляд. Також попри опис послідовних етапів APT атаки залишається невідомим спосіб детального встановлення хронології подій та пов’язування, можливо, несумісних подій між собою.

В цілому проаналізовані методи мають такі основні недоліки: відсутність схеми пошуку подій, складність в імплементації методів при аналізі, відсутність готової програмної реалізації для часткової або ж повної автоматизації аналізу APT.

Тому, зважаючи на ці недоліки, була поставлена мета створити прикладний метод аналізу вторгнень, який буде опиратися на пошук потенційних індикаторів застосування зловмисних технік, а також буде здійснювати пов’язування подій шляхом алгоритму нечіткого пошуку взаємозв’язків між техніками із ціллю встановлення хронології подій APT.

2 ПІДХОДИ ДО ВИРІШЕННЯ ПРОБЛЕМ ФІЛЬТРАЦІЇ ДАНИХ ТА ПОБУДОВИ ВЗАЄМОЗВ'ЯЗКІВ МІЖ ПОДІЯМИ

У другому розділі розглянуто базу знань MITRE ATT&CK, призначенням якої є класифікація дій зловмисників за техніками на різних етапах APT атаки. Було визначено ціль використання матриці MITRE – фільтрація подій та зменшення їх кількості шляхом пошуку застосування технік у подіях. Важливим етапом для методу аналізу було визначено пов'язання подій між собою за певними параметрами, тому у даному розділі було проведено порівняльний аналіз існуючих методів нечіткого пошуку та обрано алгоритм Вагнера-Фішера для реалізації пошуку схожих параметрів у подіях.

2.1 Матриця MITRE як підхід до фільтрації даних

У даній роботі було розглянуто Enterprise ATT&CK для побудови методу аналізу APT атак, так як саме ця матриця містить набір різних тактик та технік, які використовуються противниками на різних фазах атак, особливо дана матриця корелюється із основними етапами життєвого циклу APT атаки.

Матриця чітко описує різні дії зловмисників та уніфікує їх назви, пошук саме цих технік серед подій операційної системи надає змогу описувати діяльність противників протягом певного проміжку часу. Наявність використання технік та їх індикатори детальніше розглядається у наступному розділі.

За допомогою матриці можна вирішити одну із глобальних проблем – зменшення кількості подій для аналізу, тому практичне застосування матриці надає змогу організувати певну фільтрацію подій і водночас виділення потенційно зловмисних подій, що можуть вказувати на діяльність зловмисників в аналізованій інфраструктурі.

Наразі Enterprise ATT&CK містить 12 тактик, короткий опис яких подано у Таблиці 2.1. Дані тактики паралельно виступають в ролі етапів APT атаки.

Таблиця 2.1 – Тактики Enterprise ATT&CK

Назва тактики	Опис
Початковий доступ (Initial Access)	Противник намагається проникнути у мережу.
Виконання (Execution)	Противник намагається застосувати шкідливий код.
Постійність (Persistence)	Противник намагається утримати свою присутність.
Підвищення привілеїв (Privilege Escalation)	Противник намагається отримати дозвіл вищого рівня.
Обхід захисту (Defense Evasion)	Противник намагається уникнути його виявлення.
Доступ до облікових даних (Credential Access)	Противник намагається вкрасти імена та паролі облікових записів.
Виявлення (Discovery)	Противник намагається з'ясувати оточення у мережі.
Переміщення в мережі (Lateral Movement)	Противник намагається просуватися у мережі.
Збір даних (Collection)	Противник намагається зібрати дані, які йому потрібні.
Командування і управління (Command and Control)	Противник намагається спілкуватися з скомпрометованими системами для їх контролю.
Просочення даних (Exfiltration)	Противник намагається вкрасти дані.
Вплив (Impact)	Противник намагається маніпулювати, створювати перебої або знищувати цільові системи та дані.

Для розгляду в роботі було обрано матрицю Enterprise для операційної системи Windows, усі техніки було проаналізовано в розрізах усіх наявних тактик та визначено, які з цих технік можна реалізувати за допомогою типових утиліт даної операційної системи та внаслідок цих дій отримати події.

Внаслідок такого типу аналізу для матриці Enterprise ATT&CK для операційної системи Windows було помічено техніки, що були виконані і протестовані у тестовому середовищі (для деяких технік існували створені тести від Atomic Red Team [9], а для інших технік створювалися власні тести на основі проаналізованих атак та інформації про техніки), також залишилася частина технік для яких не було виявлено потрібних подій у подіях Windows, тому ці техніки залишилися не покритими.

Atomic Red Team це команда, яка розробила невеликі, портативні тести, відображені в рамках MITRE ATT&CK Framework. Кожен тест призначений для відображення певної техніки у розрізі тактики. Це дає спеціалістам з інформаційної безпеки надзвичайно дієвий спосіб негайно розпочати перевірку ефективності засобів захисту проти широкого спектру атак. Але дана команда описала тести не для всіх технік, а також не покривала певні аспекти технік, що були власноруч доповнені внаслідок дослідження технік MITRE ATT&CK.

Дослідження технік відбувалося шляхом аналізу відповідних атак, у яких мало місце застосування цих технік, методи застосування розглядалися та були протестовані у тестовому середовищі, внаслідок чого було отримано потрібні події Windows та визначені індикатори для цих технік.

У Таблиці 2.2 і Таблиці 2.3 подана матриця технік MITRE ATT&CK для операційної системи Windows із помітками технік, які були оброблені. Сірим кольором помічені покриті виявленими індикаторами техніки, білим – не покриті. Дані матриці було розділено на дві таблиці через великий об’єм даних, який потрібно відобразити у таблицях. Усі техніки строго відмічені у розрізі тактик, існують повторення технік у різних тактиках, так як одна і та ж техніка може по-різному бути застосована на різних етапах атаки.

Детальна інформація за індикаторами технік була описана у наступному розділі.

Основною метою використання матриці MITRE є фільтрація подій та зменшення їх кількості шляхом пошуку застосування технік у подіях на основі побудування правил на спрацювання за матрицями нижче.

Таблиця 2.2 – Матриця покриття індикаторами для Windows частина 1

Initial Access	Execution		Persistence			Privilege Escalation		Defense Evasion			
Drive-by Compromise	CMSTP	Regsvcs/Regasm	Accessibility Features	External Remote Services	Redundant Access	Access Token Manipulation	Hooking	Access Token Manipulation	Deobfuscate/Decode Files or Information	Indirect Command Execution	Regsvr32
Exploit Public-Facing Application	Command-Line Interface	Regsvr32	Account Manipulation	File System Permissions Weakness	Registry Run Keys / Startup Folder	Accessibility Features	Image File Execution Options Injection	BITS Jobs	Disabling Security Tools	Install Root Certificate	Rootkit
External Remote Services	Compiled HTML File	Rundll32	AppCert DLLs	Hidden Files and Directories	SIP and Trust Provider Hijacking	AppCert DLLs	New Service	Binary Padding	Execution Guardrails	InstallUtil	Rundll32
Hardware Additions	Control Panel Items	Scheduled Task	AppInit DLLs	Hooking	Scheduled Task	AppInit DLLs	Path Interception	Bypass User Account Control	Exploitation for Defense Evasion	Masquerading	SIP and Trust Provider Hijacking
Replication Through Removable Media	Dynamic Data Exchange	Scripting	Application Shimming	Hypervisor	Screensaver	Application Shimming	Port Monitors	CMSTP	Extra Window Memory Injection	Modify Registry	Scripting
Spearphishing Attachment	Execution through API	Service Execution	Authentication Package	Image File Execution Options Injection	Security Support Provider	Bypass User Account Control	Process Injection	Code Signing	File Deletion	Mshta	Signed Binary Proxy Execution
Spearphishing Link	Execution through Module Load	Signed Binary Proxy Execution	BITS Jobs	LSASS Driver	Service Registry Permissions Weakness	DLL Search Order Hijacking	SID-History Injection	Compile After Delivery	File Permissions Modification	NTFS File Attributes	Signed Script Proxy Execution
Spearphishing via Service	Exploitation for Client Execution	Signed Script Proxy Execution	Bootkit	Logon Scripts	Shortcut Modification	Exploitation for Privilege Escalation	Scheduled Task	Compiled HTML File	File System Logical Offsets	Network Share Connection Removal	Software Packing
Supply Chain Compromise	Graphical User Interface	Third-party Software	Browser Extensions	Modify Existing Service	System Firmware	Extra Window Memory Injection	Service Registry Permissions Weakness	Component Firmware	Group Policy Modification	Obfuscated Files or Information	Template Injection
Trusted Relationship	InstallUtil	Trusted Developer Utilities	Change Default File Association	Netsh Helper DLL	Time Providers	File System Permissions Weakness	Valid Accounts	Component Object Model Hijacking	Hidden Files and Directories	Process Doppelgänging	Timestomp
Valid Accounts	LSASS Driver	User Execution	Component Firmware	New Service	Valid Accounts		Web Shell	Control Panel Items	Image File Execution Options Injection	Process Hollowing	Trusted Developer Utilities
	Mshta	Windows Management Instrumentation	Component Object Model Hijacking	Office Application Startup	Web Shell			DCShadow	Indicator Blocking	Process Injection	Valid Accounts
	PowerShell	Windows Remote Management	Create Account	Path Interception	Windows Management Instrumentation Event Subscription			DLL Search Order Hijacking	Indicator Removal from Tools	Redundant Access	Virtualization/Sandbox Evasion
		XSL Script Processing	DLL Search Order Hijacking	Port Monitors	Winlogon Helper DLL			DLL Side-Loading	Indicator Removal on Host	Regsvcs/Regasm	Web Service
											XSL Script Processing

Таблиця 2.3 – Матриця покриття індикаторами для Windows частина 2

Credential Access	Discovery		Lateral Movement	Collection	Command and Control		Exfiltration	Impact
Account Manipulation	Account Discovery	Process Discovery	Application Deployment Software	Audio Capture	Commonly Used Port	Multi-hop Proxy	Automated Exfiltration	Data Destruction
Brute Force	Application Window Discovery	Query Registry	Distributed Component Object Model	Automated Collection	Communication Through Removable Media	Multiband Communication	Data Compressed	Data Encrypted for Impact
Credential Dumping	Browser Bookmark Discovery	Remote System Discovery	Exploitation of Remote Services	Clipboard Data	Connection Proxy	Multilayer Encryption	Data Encrypted	Defacement
Credentials in Files	Domain Trust Discovery	Security Software Discovery	Logon Scripts	Data Staged	Custom Command and Control Protocol	Remote Access Tools	Data Transfer Size Limits	Disk Content Wipe
Credentials in Registry	File and Directory Discovery	System Information Discovery	Pass the Hash	Data from Information Repositories	Custom Cryptographic Protocol	Remote File Copy	Exfiltration Over Alternative Protocol	Disk Structure Wipe
Exploitation for Credential Access	Network Service Scanning	System Network Configuration Discovery	Pass the Ticket	Data from Local System	Data Encoding	Standard Application Layer Protocol	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Forced Authentication	Network Share Discovery	System Network Connections Discovery	Remote Desktop Protocol	Data from Network Shared Drive	Data Obfuscation	Standard Cryptographic Protocol	Exfiltration Over Other Network Medium	Firmware Corruption
Hooking	Network Sniffing	System Owner/User Discovery	Remote File Copy	Data from Removable Media	Domain Fronting	Standard Non-Application Layer Protocol	Exfiltration Over Physical Medium	Inhibit System Recovery
Input Capture	Password Policy Discovery	System Service Discovery	Remote Services	Email Collection	Domain Generation Algorithms	Uncommonly Used Port	Scheduled Transfer	Network Denial of Service
Input Prompt	Peripheral Device Discovery	System Time Discovery	Replication Through Removable Media	Input Capture	Fallback Channels	Web Service		Resource Hijacking
Kerberoasting	Permission Groups Discovery	Virtualization/Sandbox Evasion	Shared Webroot	Man in the Browser	Multi-Stage Channels			Runtime Data Manipulation
LLMNR/NBT-NS Poisoning and Relay			Taint Shared Content	Screen Capture				Service Stop
Network Sniffing			Third-party Software	Video Capture				Stored Data Manipulation
Password Filter DLL			Windows Admin Shares					Transmitted Data Manipulation
Private Keys			Windows Remote Management					
Two-Factor Authentication Interception								

2.2 Побудова взаємозв’язків подій за допомогою нечіткого пошуку

Під час аналізу процесів та їх діяльності в інформаційних подіях будь-якої операційної системи постає питання зв’язування цих подій між собою. Якщо відбувається розгляд подій у розрізі одного пристрою, то пошук взаємозв’язків між процесами можна здійснювати за допомогою звичайного пошуку, де вихідний процес однієї події буде викликати цільовий процес в іншій події. Але під час аналізу події від декількох пристроїв створює проблему, що полягає у

можливих відмінностях у місцезнаходженнях процесів (наприклад, на двох пристроях різні змінні середовища для однакових програмних засобів), у версіях процесів, у навмисно змінених назвах процесів. Тому альтернативний пошук процесів за допомогою хешів може виявитися неефективним, так як різні хости можуть мати процеси різних версій, у яких повністю відрізняється хеш, а також шлях до процесу може бути інакшим. У цьому випадку зв'язування процесів за назвою, хешем чи шляхом до процесу не може бути використаним, так як великий відсоток пов'язаних процесів буде втрачено під час аналізу. Також важливим моментом є порівняння параметрів командного рядка, у яких можуть бути відмінності у регістрі буквенних знаків, кількості символів, місцезнаходження процесу, який викликається. У даному випадку також піднімається питання, яким чином можна класифікувати ці дві події пов'язаними чи схожими, якщо за допомогою прямого зрівняння ці події будуть різними.

Наразі для здійснення пошуку, що дозволяє наявність деяких відмінностей у ключовому слові та у знайденій інформації, використовується тип пошуку, який називається нечітким.

Нечіткий пошук – це пошук інформації, при якому відбувається зіставлення інформації із заданим прикладом пошуку або близьким значенням до цього прикладу[6]. Алгоритми нечіткого пошуку використовуються у більшості сучасних пошукових системах (наприклад, для перевірки орфографії). Формальне визначення алгоритму: нехай $\sum$ – скінчена множина (алфавіт) розміру $|\sum| = \sigma$. Нехай $T \in \sum^*$ – текст довжиною $n = |T|$. Нехай $P \in \sum^*$ – приклад довжиною $m = |P|$. Нехай $k \in N$ – максимально дозволена кількість помилок. Нехай $d : \sum^* \times \sum^* \rightarrow N$ – функція відстані. Тоді задача алгоритму: дані T, P, k і d(.), які повертають множини усіх позицій тексту j таких, що існує i таке, що $d(P, Ti\ldots j) \leq k$ (2). Інакше кажучи, задачею алгоритму є пошук за заданим словом у тексті розміру n усіх слів, які співпадають із цим словом із врахуванням k можливих неточностей.

Відстань Левенштейна між двома рядками – це мінімальна кількість операцій вставки одного символу, видалення одного символу і заміни одного

символу на інший, необхідних для перетворення одного рядка в інший. Рекурентна формула, що описує відстань Левенштейна (2.1):

$$d(S_1, S_2) = D(M, N), \text{де}$$

$$D(i,j) = \begin{cases} 0, i = 0, j = 0 \\ i, j = 0, i > 0 \\ j, i = 0, j > 0 \\ D(i-1, j-1), S_1[i] = S_2[j] \\ \min\big(D(i, j-1), D(i-1, j), D(i-1, j-1)\big), j > 0, i > 0, S_1[i] \neq S_2[j] \end{cases} \quad (2.1)$$

2.3 Порівняння методів нечіткого пошуку

Перший алгоритм, який було розглянуто, це алгоритм Soundex [11, 12]. Це один із алгоритмів порівняння двох рядків за їх звучанням. Він встановлює однаковий індекс для рядків, що мають схоже звучання у мові відповідно до заданої таблиці схожих за звучанням символів і їх поєднань. Однак, він має істотний недолік: цей алгоритм прив'язаний до мови, на якому написані аналізовані рядки. Алгоритм використовується в даний час в основному в англомовному середовищі, для якої подібна таблиця вже існує.

Наступний з розглянутих алгоритмів – алгоритм розширення вибірки [13]. Даний алгоритм часто застосовується в системах перевірки орфографії. Він заснований на зведенні задачі по нечіткому пошуку до задачі про точний пошук. Даний метод передбачає побудову найбільш ймовірних «неправильних» варіантів пошукового шаблону.

Основна перевага даного алгоритму полягає в легкості його модифікації для генерації «помилкових» варіантів за довільними правилами. У алгоритму є і недоліки, головний з яких – велика кількість перевірок для слів суттєвої довжини, оскільки з них можна отримати багато «помилкових» слів.

Широко відомий також алгоритм на основі коду Хеммінга, який застосовується при кодуванні і декодуванні даних. Лінійні коди, як правило, добре справляються з рідкісними і великими помилками. Однак, їх ефективність

при порівнянні слів з частими, але невеликими помилками, досить низька. В даному алгоритмі також присутні додаткові витрати на кодування інформації.

Алгоритм Bitap (також відомий як Shift-Or або Baeza-Yates-Gonnet) і різні його модифікації найбільш часто використовуються для нечіткого пошуку без індексації [14]. Початкова версія алгоритму враховувала тільки заміну символів, і по суті обчислювала відстань Хеммінга. Але наступна модифікацію цього алгоритму вже містила функцію обчислення відстані Левенштейна, і в подальшому стала основою для першої версії Unix утиліти agrep. Висока швидкість роботи цього алгоритму зумовлена бітовим паралелізмом обчислень: за одну операцію можливо провести обчислення над 32 і більше бітами одночасно. Але використання типів великих розмірностей уповільнює роботу алгоритму, що є його недоліком.

Алгоритм Вагнера-Фішера [15] дозволяє для двох рядків знайти відстань Левенштейна. Даний алгоритм має ряд значних переваг перед усіма описаними вище, а саме: відносно невисоку складність реалізації, можливість якісного порівняння схожості більш ніж двох рядків, кілька варіантів реалізації, які можна використовувати в залежності від конфігурації системи, універсальність для всіляких алфавітів. Також у даного алгоритму існує одна цікава модифікація, яка дозволяє знаходити відстань Дамерау-Левенштейна [16]. У ньому до операцій вставки, видалення і заміни символів, визначених в відстані Левенштейна, додана операція транспозиції (перестановки) символів. Фредерік Дамерау показав, що 80% помилок при наборі тексту людиною є транспозиція.

У ході порівняння наведених алгоритмів було створено порівняльну характеристику, яка наведена у Таблиці 2.4. У даній таблиці було порівняно алгоритми нечіткого пошуку за обраними основними параметрами: універсальність алгоритми відносно алфавіту (мови) вхідних даних, які будуть порівнюватися, кодування даних має на увазі – перетворення даних за допомогою певного коду в інший формат представлення, що зменшує швидкість відпрацювання алгоритму, обробка рядків більше 32 біт означає можливість обробки великих за об’ємом рядків (наприклад, у випадку порівняння шляхів

процесів у події), складність програмної реалізації – наскільки швидким та трудомістким процесом є реалізація алгоритму на будь-якій мові програмування, операції вставки, заміни, видалення та транспозиції передбачає відповідь, чи підтримує поточний алгоритм дану операцію, наявність вагових коефіцієнтів для операцій означає можливість ранжування операцій для встановлення їх важливості для порівняння вхідних даних. За умовними позначеннями у таблиці: «+» – якщо даний параметр присутній у методі, «-» – якщо даний параметр відсутній у методі.

Таблиця 2.4 – Порівняння методів нечіткого пошуку

Параметри порівняння	Алгоритми (методи)				
	Soundex	Алгоритм розширення вибірки	Алгоритм на основі коду Хеммінга	Алгоритм Bitap	Алгоритм Вагнера-Фішера
Універсальність відносно алфавіту вхідних даних	-	+	+	+	+
Відсутність кодування даних	+	+	-	+	+
Обробка рядків більше 32 біт	-	-	+	-	+
Складність програмної реалізації середня чи низька	-	+	-	+	+
Операції вставки	-	-	-	+	+
Операції видалення	-	-	-	+	+
Операції заміни	-	-	+	+	+

Кінець таблиці 2.4

Параметри порівняння	Алгоритми (методи)				
	Soundex	Алгоритм розширення вибірки	Алгоритм на основі коду Хеммінга	Алгоритм Bitap	Алгоритм Вагнера-Фішера
Операції транспозиції	-	-	-	-	+
Вагові коефіцієнти для операцій	-	-	-	-	+
Рейтинг алгоритму	***1***	***3***	***3***	***6***	***9***

З наведеного вище аналізу відомих методів нечіткого пошуку рядків було обрано алгоритм Вагнера-Фішера із огляду на його легкість у програмній реалізації, наявність вагових коефіцієнтів для 4 типів операцій, можливість обробки великих рядків та універсальність відносно алфавіту вхідних даних. Даний алгоритм ліг в основу розробленого методу пошуку взаємозв'язків між використаними техніками MITRE для побудови хронології подій APT атаки.

2.4 Основні поняття та опис алгоритму Вагнера-Фішера

Для відстані Левенштейна наступні висловлення є справедливими:

- $d(S_1,S_2) \geqslant ||S_1|-|S_2||$,
- $d(S_1,S_2) \leqslant \max(|S_1|,|S_2|)$,
- $d(S_1,S_2)=0 \Leftrightarrow S_1=S_2$,

де $d(S_1,S_2)$— відстань Левенштейна між рядками S_1 и S_2, а $|S|$ — довжина рядка S.

Відстань Левенштейна є метрикою. Для того, щоб довести це ствердження, достатньо довести, що виконується нерівність трикутника: $d(S_1,S_3) \leqslant d(S_1,S_2)+d(S_2,S_3)$.

Нехай $d(S_1,S_3)=x$, $d(S_1,S_2)=y$, $d(S_2,S_3)=z$. x — найменша відстань Левенштейна від S_1 до S_3, y — найменша відстань Левенштейна від S_1 до S_2, а z — найменша відстань Левенштейна від S_2 до S_3. y+z — деяка відстань від S_1 до S_3. В інших випадках $d(S_1,S_3) < d(S_1,S_2)+d(S_2,S_3)$. Із цього виходить, що нерівність трикутника виконується.

Для даного алгоритму також вводиться таке поняття, як ціна операцій. Ціна операцій може залежати від виду операцій (вставка, видалення, заміна) та/або від самих типів символів, що відображає різну імовірність різних помилок при введенні тексту. У загальному випадку:

- w(a, b) — ціна заміни символу a на символ b;
- w(ε, b) — ціна вставки символу b;
- w(a, ε) — ціна видалення символу a.

Для вирішення задачі про редакційну відстань необхідно знайти послідовність замін, що мінімізує сумарну ціну. Відстань Левенштейна є частковим випадком цієї задачі при:

- w(a, a)=0;
- w(a, b)=1 при $a \neq b$;
- w(ε, b)=1;
- w(a, ε)=1;

де ε — порожня послідовність.

Як цей частковий випадок, так і задачу для довільних w, вирішує алгоритм Вагнера-Фішера, який наведено нижче. При цьому, вважатимемо, що усі w невід’ємні, і має місце правило трикутника: якщо дві послідовні операції можна замінити однією, то ця дія не погіршує загальну ціну (наприклад, замінити символ x на y, а потім із y на z не краще, чим відразу x на z).

Перед описом формули важливим зауваженням є те, що для неї має місце нумерація рядків із першого, як прийнято в математиці, а не з нульового, як це прийнято в програмуванні.

Нехай S_1 і S_2 — два рядка (довжиною M і N відповідно) над деяким алфавітом, тоді редакційна відстань $d(S_1,S_2)$ може бути підрахована за наступною рекурентною формулою (2.2):

$$d(S_1, S_2) = D(M, N), \text{де}$$

$$D(i,j) = \begin{cases} 0, i = 0, j = 0 \\ i * deleteCost, j = 0, i > 0 \\ j * insertCost, i = 0, j > 0 \\ D(i-1, j-1), S_1[i] = S_2[j] \\ \min \begin{pmatrix} D(i, j-1) + insertCost, \\ D(i-1, j) + deleteCost, \\ D(i-1, j-1) + replaceCost \end{pmatrix}, j > 0, i > 0, S_1[i] \neq S_2[j] \end{cases} \quad (2.2)$$

Доведення. D(i, j) – відстань між префіксами рядків: першими *i* символами рядку S_1 і першими *j* символами рядка S_2. Очевидно, що для двох порожніх рядків редакційна відстань буде рівна нулю. Так само очевидно, що для того, щоб отримати порожній рядок із рядка довжиною *i*, потрібно зробити *i* операцій видалення, а щоб отримати рядок довжиною *j* із порожнього рядка, то потрібно зробити *j* операцій вставки.

Для розгляду залишається нетривіальний випадок, коли обидва рядка не порожні.

В оптимальній послідовності операцій їх можна довільно міняти місцями. Розглянемо дві послідовні операції:

- Дві заміни одного і того ж символу – не оптимально (якщо замінено x на y, потім y на z, вигідніше було відразу замінити x на z).
- Дві заміни різних символів можна міняти місцями.
- Два стирання або дві вставки можна міняти місцями.
- Вставка символу з його подальшим стиранням – не оптимально (можна їх обидві скасувати).
- Стирання і вставку різних символів можна міняти місцями.
- Вставка символу з його подальшою заміною – не оптимально (зайва заміна).

- Вставка символу і заміна іншого символу міняються місцями.
- Заміна символу з його подальшим стиранням – не оптимально (зайва заміна).
- Стирання символу і заміна іншого символу міняються місцями.

Нехай S_1 закінчується на символ *a*, S_2 закінчується на символ *b*. Є три варіанти:

- Символ *a*, на який закінчується S_1, в якийсь момент був стертий. Зробимо це стирання першою операцією. Тоді стерли символ a, після чого перетворили перші i-1 символів S_1 в S_2 (на що було потрібно D (i-1, j) операцій), тоді всього було потрібно D (i-1, j) +1 операцій.
- Символ *b*, на який закінчується S_2, в якийсь момент був доданий. Зробимо це додавання останньою операцією. S_1 було перетворено у перші j-1 символів S_2, після чого було додано символ *b*. Аналогічно попередньому випадку, треба було D (i, j-1) +1 операцій.
- Обидва попередніх твердження невірні. Якщо додавали символи праворуч від фінального *a*, то щоб зробити останнім символом *b*, потрібно в якийсь момент додати його (але тоді твердження 2 було б правильним), або замінити на нього один з цих доданих символів (що теж неможливо, тому що додавання символу з його подальшою заміною не оптимально). Значить, символів праворуч від фінального *a* ми не додавали. Самого фінального *a* не стирали, оскільки твердження 1 невірне. Значить, єдиний спосіб зміни останнього символу – його заміна. Замінювати його 2 або більше разів не оптимально. Значить:
  - Якщо $a = b$, останній символ не змінювали. Оскільки його теж не стирали і не приписували нічого праворуч від нього, він не впливав на наші дії, і, значить, було виконано D (i-1, j-1) операцій.
  - Якщо $a \neq b$, останній символ було змінено один раз. Зробимо цю заміну першою. Надалі, аналогічно попередньому випадку, потрібно виконати D (i-1, j-1) операцій, тому всього буде потрібно D (i-1, j-1) +1 операцій.

Суттю алгоритму Вагнера-Фішера є знаходження найкоротшої відстані, для чого необхідно обчислити матрицю D за формулою (2.2). Її можна обраховувати як за рядками, так і за стовпцями.

Псевдокод алгоритму, написаний для довільних цін заміни, вставок та видалення (необхідно мати на увазі, що елементи нумеруються з 1). Псевдокод нижче вирішує простий частковий випадок, коли вставка символу, видання символу чи заміна одного символу на інший коштують однаково для будь-яких символів:

```
int levensteinInstruction (String s1, String s2, int InsertCost, int DeleteCost, int ReplaceCost):
        D[0][0] = 0
        for j = 1 to N
                D[0][j] = D[0][j - 1] + InsertCost
        for i = 1 to M
                D[i][0] = D[i - 1][0] + DeleteCost
                for j = 1 to N
                        if S1[i] != S2[j]
                                D[i][j] = min(D[i - 1][j] + DeleteCost,
                                D[i][j - 1] + InsertCost,
                                D[i - 1][j - 1] + ReplaceCost)
                        else
                                D[i][j] = D[i - 1][j - 1]
return D[M][N]
```

Алгоритм, у вигляді описаному вище, потребує $\Theta(M \cdot N)$ операцій і таку ж пам'ять, хоча якщо потрібна тільки відстань, то пам'ять можна зменшити до $\Theta(N)$.

Висновки до розділу 2

У розділі було подано короткий опис тактик MITRE та показано покриття визначеними індикаторами у ході проведеної роботи. Важливістю застосування матриці MITRE є застосування її для початкової фільтрації подій, виділення нормальних подій та потенційно зловмисних, які були класифіковані як дії використання технік.

Також було визначено, що необхідним у аналізі подій для визначення APT атаки є пов’язування різних процесів між собою. Але дана діяльність має проблему, що полягає у можливих відмінностях у місцезнаходженнях процесів, у їх версіях та у потенційно навмисно змінених назвах процесів, що робить пов’язування процесів за допомогою хешів чи за шляхом до процесу неефективними підходами. Рішенням цього питання було обрано використання алгоритму нечіткого пошуку.

У ході порівняльного аналізу алгоритмів нечіткого пошуку за обраними критеріями було вирішено використовувати у методі аналізу APT атак алгоритм Вагнера-Фішера, так як він має невисоку складність реалізації, можливість якісного порівняння схожості більш ніж двох рядків, універсальність для різних алфавітів, легкість у програмній реалізації, наявність вагових коефіцієнтів для 4 типів операцій, можливість обробки великих рядків.

3 РОЗРОБКА МЕТОДУ АНАЛІЗУ APT АТАК НА ОСНОВІ ТЕХНІК MITRE ТА АЛГОРИТМУ ВАГНЕРА-ФІШЕРА

У третьому розділі описано розроблений метод аналізу APT атак та його основні етапи. Було представлено виявлені індикатори технік MITRE, які призначені для першого етапу методу – фільтрації подій. Також описано програмну реалізацію, яка покриває інші етапи методу аналізу, зокрема містить алгоритм нечіткого пошуку та підходи до побудови послідовності подій та хронології виявленої APT атаки.

3.1 Опис розробленого методу аналізу APT

На основі розглянутих матриці MITRE та обраного алгоритму нечіткого пошуку Вагнера-Фішера було сформовано метод аналізу APT атак. Основними етапами даного методу є:

1. Фільтрація даних (подій від Windows Sysmon та Powershell) за допомогою виявлення застосування технік MITRE.
2. Застосування алгоритму нечіткого пошуку до отриманих даних для побудови взаємозв’язків між процесами.
3. Побудова послідовності застосування технік MITRE та співставлення із життєвим циклом APT атаки.
4. Визначення хронології подій APT атаки.

Першим етапом є фільтрація даних, яка є одним із найголовніших етапів. Етап фільтрації полягає у відсіюванні даних за допомогою пошуку індикаторів застосування технік у подіях. Відсіювання даних набагато спрощує пошук потенційно зловмисних діянь серед нормальних і легітимних подій в інформаційній системі. В середньому кількість подій Windows Sysmon та Powershell в організації із 50 хостів складає 171644 події, в той час як кількість

подій, що підлягають умовам застосування технік MITRE – 55 подій (значення отримане за період від 01.06.2019 до 01.11.2019). Тобто різниця у кількості даних для опрацювання колосальна – із застосуванням фільтрації кількість подій для аналізу скорочується у 3120 разів. Тому етап фільтрації даного методу – це реалізація та пошук подій застосованих технік на основі отриманих індикаторів.

На Рис.3.1 показано об’єм трафіку від Windows Sysmon та Powershell в організації із 50 хостів.

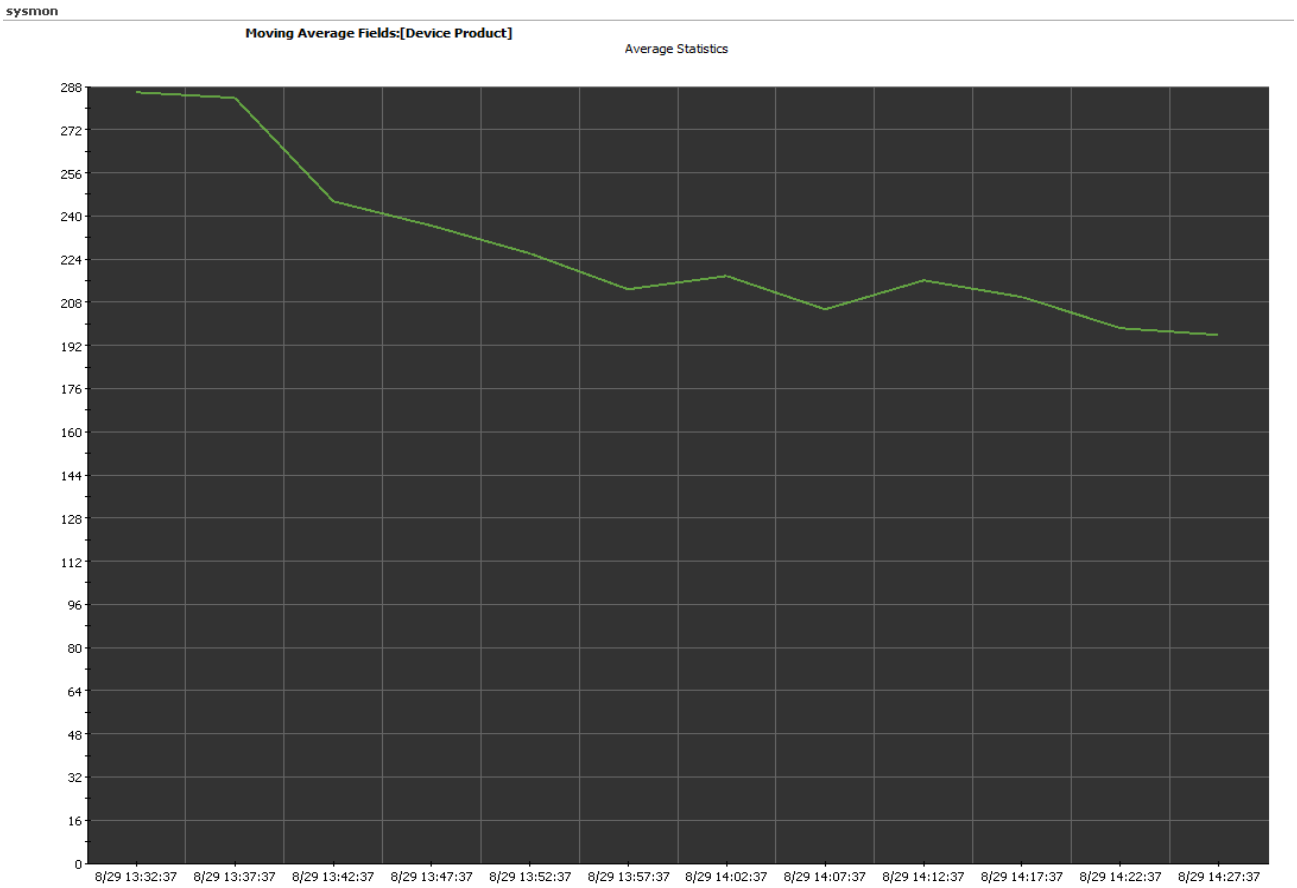

Рисунок 3.1 – Об’єм трафіку від Windows Sysmon та Powershell

Етап застосування алгоритму нечіткого пошуку до отриманих даних полягає в обчисленні відстані Левенштейна для зрівнюваних полів у різних подіях, зокрема для цільових процесів у подіях. Ця операція націлена на пошук зв’язків між подіями – який процес ініціював запуск іншого. Застосування саме такого алгоритму обґрунтовано тим, що у процесів на різних хостах можуть бути відмінності у назвах процесів – нижній чи верхній регістр; різниця у папках, де

знаходиться процес, що залежить від змінних середовища в операційній системі; спроби зловмисників зміни назв процесів (наприклад, mimikatz та mimikittenz).

Наступним етапом є побудова послідовності застосування технік MITRE та співставлення із життєвим циклом APT атаки. Так як ряд технік можуть бути застосованими тільки на певних етапах APT, то є сенс у тому, щоб упорядкувати застосовані техніки згідно із життєвим циклом атаки, щоб відсіяти фальш-позитивні спрацювання. Заключним етапом є побудова хронології подій APT атаки, де описовими пунктами слугують застосовані техніки.

3.2 Етап фільтрації подій

Фільтрація даних була створена за допомогою застосування правил у SIEM ArcSight у даній роботі. Для написання даних правил було розглянуто техніки за кожним напрямом MITRE, частина технік була опрацьована командою Atomic Red Team, які пропонують свої тести для перевірки систем безпеки на рівень захищеності, а також були застосовані власні тести на основі аналізу атак. Дані тести були відтворені на двох віртуальних машинах Windows 10 для отримання подій від Sysmon [21] та Powershell [22].

Sysmon має достатньо різних подій, які містять спеціальний ідентифікатор під кожен тип події. У ході розгляду подій бралися до уваги такі ідентифікатори:

1 – Створено новий процес;

3 – Ініційована Інтернет комунікація;

7 – Завантажено процесом модуль dll;

11 – Створено новий файл;

12 – Об’єкт реєстру був видалений/створений;

13 – Задано нове значення в реєстр;

15 – Створено файловий потік.

22 – Виявлено DNS запит.

Деякі техніки MITRE були частково покриті тестами, не залишали подій в Sysmon або взагалі не були покриті тестами від Atomic Red Team, тому проводився додатковий аналіз атак, вказаних в якості прикладів на сторінках опису технік. В результаті проведення низки робіт із тестування технік у віртуальному середовищі були отримані індикатори компрометації (IoC) для кожної із протестованих технік.Так як дані індикатори можуть бути застосовані в інших SIEM системах або в аналізаторах подій, то назви змінних універсальні і зрозумілі, не орієнтовані під певний сервіс:

- TargetProcessName (TPN) – назва цільового процесу;
- SysmonEventID (SEID) – ідентифікатор події в Sysmon;
- PE (PE) – подія із powershell;
- Command Line (CMD:) – командний рядок;
- Source Process Name (SPN:) – назва процесу джерела;
- Parent Process Cmd Line (PPCMD:) – командний рядок батьківського процесу;
- Parent Process Path (PPP:) – місцезнаходження батьківського процесу;
- File Path (FP:) – місцезнаходження процесу;
- Loaded Module Name (DLL:) – завантажена бібліотека із розширенням .dll;
- Request Method (RM:) – метод запиту;
- Target Object Path (TOP:) – папка цільового об’єкту;
- «–» – пусте значення.

Під час реалізації етапу фільтрації було написано правила (Rules) у SIEM системі. Приклад умов спрацювання правила показано на Рис.3.2.

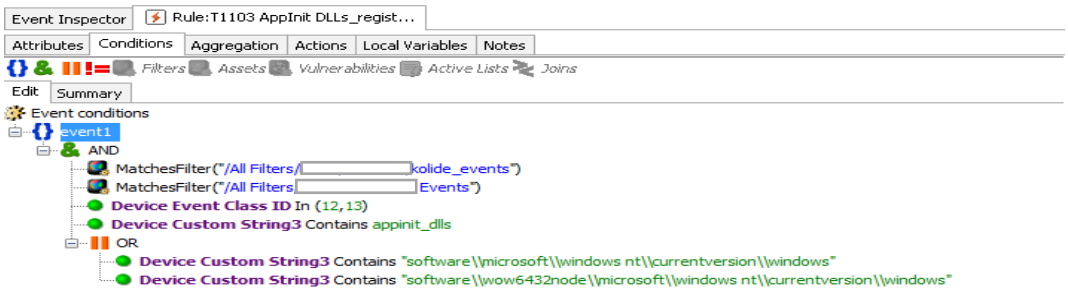

Рисунок 3.2 – Приклад умов правила

Загалом було створено 209 правил для 124 технік (більшість технік здійсненні різними шляхами, тому відбувався розподіл правил за типами реалізації техніки та типом подій).

Типи подій у правилах об’єднувалися в основні групи: registry event (події Sysmon 12, 13), process event (події Sysmon 1, 11, 15), network event (події Sysmon 3, 22), module loaded (події Sysmon 7), powershell event (події Powershell). За типами подій було проведено розділення правил і отримано: 36 правил на основі registry event, 114 правил на основі process event, 19 правил на основі network event, 5 правил на основі module loaded, 34 правила на основі powershell event.

3.3 Опис індикаторів технік для фільтрації подій

Тактика Початковий доступ (Initial Access) передбачає, що зловмисник має на увазі проникнення до цільової системи шляхом використання слабких місць та вразливостей на загальнодоступних серверах, надсилання зловмисних посилань чи файлів через пошту, а також через використання справжніх облікових записів, що були викрадені раніше.

Всього тактика налічує 11 технік, але під час їх аналізу виникла складність у тому, що більшість технік підлягає такій реалізації, яка легко маскується під звичайну діяльність і подій з sysmon та powershell недостатньо для її виявлення. Всього тестів існує для 2 технік, один із них власний, інший від Atomic Red Team. Із решти 9 технік цілих 5 не мають представлення у подіях Windows у будь-якому розділі, а залишають сліди лише в подіях мережевих засобів. Техніка, що була запропонована від Atomic, - Spearfishing attachment, - була протестована, але слідів її використання знайдено не було, тому у розрізі даної тактики було протестована власними тестами інша техніка – Зовнішні віддалені сервіси. Особливість даної техніки в тому, щоб відслідковувати застосування сервісів віддаленого доступу за межами робочого часу. Опис її індикаторів представлені нижче у Таблиці 3.1.

Таблиця 3.1 – Тактика Початковий доступ (Initial Access) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1133 External Remote Services	–	FP: OR Contains "citrix.exe", “vpn” Time is before and after work hours	1,11,15,3, 22

Тактика «Виконання (Execution)» описує застосування зловмисниками доступних їм засобів і методів віддаленого і локального виконання різних команд, сценаріїв і виконуваних файлів у цільовій системі, які були доставлені в неї на попередньому етапі. Всього дана тактика містить 27 відомих технік для застосування, із яких 6 технік залишилися без покриття тестами.

Із представленої нижче Таблиці №2 було застосовано тести від Atomic Red Team для 18 технік (із них 8 технік доповнені індикаторами на основі проаналізованих атак), 4 техніки були повністю покриті власними тестами. Із виявленими індикаторами можна ознайомитися у Таблиці 3.2 нижче.

Таблиця 3.2 – Тактика Виконання (Execution) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1191 CMSTP	cmstp.exe	CMD: OR Contains " /s", “.bat”	1, 11, 15, 3
	cmstp.exe	–	1, 11, 15
	cmstp.exe	–	3
T1059 Command-Line Interface	cmd.exe	SPN: NOT Is “explorer.exe” FP: NOT Is NULL AND NOT Is “"c:\\windows\\system32\\cmd.exe"”, “"c:\\windows\\syswow64\\cmd.exe"”	1
T1223 Compiled HTML File	hh.exe	CMD: OR Contains "hh.exe", "hh", “.bat”	1, 11, 15
	hh.exe	–	1, 11, 15
	hh.exe	–	3
T1173 Dynamic Data Exchange	–	TOP: Contains “AllowDDE” AND Contains “2” TOP: Contains “ HKEY_CURRENT_USER\Software\Microsoft\Office\” AND Contains “ \Word\Security”	12, 13
T1118 InstallUtil	installutil.exe	CMD: OR Contains “installutil.exe”, “installutil”, “.bat”	1, 11, 15
	–	SPN: installutil.exe	1, 11, 15

Продовження таблиці 3.2

Техніка	TPN	Інші змінні	SEID / PE
T1177 LSASS Driver	–	CMD: NOT IS “dword (0x00000001)” PPCMD: EndsWith “SYSTEM\\CurrentControlSet\\Control\\Lsa\\RunAs PPL” FP: EndsWith “lsass.exe”	12, 13
	–	FP: EndsWith “lsass.exe”	7
T1170 Mshta	mshta.exe	CMD: OR Contains "mshta.exe", “mshta”, “.bat”	1, 11, 15
	mshta.exe	CMD: Contains " javascript:"	1, 11, 15
	–	SPN: mshta.exe	1, 11, 15
	mshta.exe	–	3
T1086 Powershell	–	CMD: OR Contains: “-noprofile”, “-windowstyle hidden”, “-executionpolicy bypass”, “UTF8”, “Base64”, “-nop”, “-w hidden”, “-e”, “-EncodedCommand”, “Invoke-Expression”	PE
T1121 Regsvcs/Regasm	regsvcs.exe/regasm.exe	CMD: OR Contains: “regsvcs.exe ”, “regsvcs”, “regasm.exe ”, “regasm”, “.bat”	1, 11, 15
	regsvcs.exe/regasm.exe	CMD: false	7
T1117 Regsvr32	regsvr32.exe	CMD: OR Contains: “regsvr32.exe ”, “regsvr32”, “.bat” RM: NOT Is “trusted”	1, 11, 15
	!=regsvr32.exe	SPN: regsvr32.exe	1, 11, 15
	regsvr32.exe	–	3
	regsvr32.exe	CMD: Contains “/i:”,	1, 11, 15, 3
T1085 Rundll32	rundll32.exe	CMD: Contains " javascript:"	1, 11, 15
	rundll32.exe	–	3
T1053 Scheduled Task	at.exe	CMD: Contains “/interactive"	1, 11, 15
	–	CMD: AND Contains: “SCHTASKS”, “/interactive", “/Create”, “/ST”, “/TR”	1, 11, 15
T1064 Scripting	wmic.exe	CMD: Contains “.xsl"	1, 11, 15
	rundll32.exe	CMD: Contains “.sct"	1, 11, 15
	wscript.exe /cscript.exe	CMD: OR Contains: “.vbs”, “wsc”, “.wsf”, “.wsh”, “.vbe”	1, 11, 15
	–	CMD: OR Contains “.psd1”, “.psm1”, “.ps1”	PE
T1035 Service Execution	sc.exe	CMD: OR Contains “create”, “delete”, “config”, “.bat”	1, 11, 15
T1216 Signed Script Proxy Execution	cscript.exe	–	1, 15
	cscript.exe	–	3
	cscript.exe	CMD: Contains “pubprn.vbs”	1, 15
T1127 Trusted Developer Utilities	msbuild.exe/ dnx.exe/rcsi.exe/windbg.exe/cdb.exe	CMD: OR Contains “”, “.bat”	1, 11, 15
	–	SPN: msbuild.exe, dnx.exe, rcsi.exe, windbg.exe, cdb.exe)	1, 15

Кінець таблиці 3.2

Техніка	TPN	Інші змінні	SEID / PE
T1047 Windows Management Instrumentation	–	SPN: wmic.exe	1, 15
	wmic.exe	CMD: OR Contains “useraccount”, “.bat”, “process”, “qfe”, “/node”	1, 11, 15
	wmic.exe	–	3
T1028 Windows Remote Management	–	CMD: Contains “Enable-PSRemoting”	PE
	powershell.exe	CMD: AND Contains “createinstance”, “MMC20.application”, “gettypefromprogid”	PE
	wmic.exe	SPN: cmd.exe CMD: Contains “process call create”	1, 15
	–	CMD: Contains “gettypefromprogid”	PE
T1220 XSL Script Processing	msxsl.exe	CMD: OR Contains “msxsl.exe”, “msxsl”, “.bat”	1, 11, 15
	–	SPN: msxsl.exe	1, 15
	msxsl.exe	–	3
	wmic.exe	CMD: OR Contains “xsl”, “/format:”, “.bat”	1, 11, 15

Тактика Постійність (Persistence) складається із технік, які супротивники використовують, щоб утримати доступ до систем під час перезавантаження, зміни облікових даних та інших перебоїв, які можуть скасувати їх доступ до цільової системи. Методи, що застосовуються для цієї тактики, включають будь-які зміни доступу, дій чи конфігурацій, які дозволяють їм підтримувати свою присутність у системах.

Дана тактика містить 43 техніки в цілому. Із них було не покрито жодними тестами 8 технік (із яких 3 залишають події тільки для мережевих засобів і пристроїв). Всього 33 техніки були покриті тестами і в подальшому правилами на спрацювання за індикаторами.

Якщо відділити ті техніки, які зустрічалися раніше в інших тактиках, то команда Atomic Red Team пропонує тести для 25 технік, із них 1 техніка не залишила слідів у відповідних подіях, 3 техніки доповнені виявленими з атак індикаторами. Також власноруч було запропоновано тести для 9 інших технік. Із виявленими індикаторами за техніками тактики Постійність можна ознайомитися у Таблиці 3.3 нижче.

Таблиця 3.3 – Тактика Постійність (Persistence) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1015 Accessibility Features	cmstp.exe	TOP: Contains “debugger” AND TOP: OR Contains “osk”, “sethc”, “utilman”, “magnify”, “narrator”, “displayswitch”, “atbroker” AND TOP: OR Contains “\software\microsoft\windows nt\currentversion\image file execution options\” , “\software\wow6432node\microsoft\windows nt\currentversion\image file execution options\”	12, 13
T1098 Account Manipulation	–	CMD: OR Contains: (AND Contains “$member”, “-like”, “Rename-LocalUser -Name”, “admin”), (AND Contains “Get-CimInstance -ClassName”, “-Filter”, “Rename-LocalUser -Name”)	PE
T1182 AppCert DLLs	–	TOP: Contains “System\CurrentControlSet\Control\Session Manager\KnownDLLs”	12, 13
T1103 AppInit DLLs	–	TOP: Contains “appinit_dlls” TOP: OR Contains “software\microsoft\windows nt\currentversion\windows”, “software\wow6432node\microsoft\windows nt\currentversion\windows”	12, 13
T1138 Application Shimming	sdbinst.exe	CMD: OR Contains “sdbinst.exe”, “.bat”, “sdbinst”	1, 11, 15
	–	TOP: Contains “appinit_dlls” TOP: OR Contains “hklm\software\microsoft\windows nt\currentversion\appcompatflags\installedsdb”, “hklm\software\microsoft\windows nt\currentversion\appcompatflags\custom”	12, 13
T1131 Authentication Package	reg.exe	CMD: Contains “HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Authentication Packages”	12, 13
T1197 BITS Jobs	bitsadmin.exe	CMD: OR Contains “bitsadmin.exe ”, “.bat”, “bitsadmin”	1, 11, 15
	–	CMD: Contains “Start-BitsTransfer”	PE
T1176 Browser Extensions	–	FP: “c:\program files (x86)\google\chrome\application\chrome.exe” TOP: Contains “AppData\Local\Google\Chrome\User Data\Default\Extensions”	1, 22, 3, 15
T1042 Change Default File Association	cmd.exe	CMD: Contains “assoc”	1, 15, 11

Продовження таблиці 3.3

Техніка	TPN	Інші змінні	SEID / PE
T1122 Component Object Model Hijacking	–	AND: TOP: Contains “\inprocserver32” TOP: OR Contains “hkcu\software\classes\clsid\”, “hklm\software\classes\clsid\”, “hkcr\clsid\” (OR CMD: Contains “scrobj”, (AND CMD: Contains “:\\”, PM: NOT trusted,not available)))	13
T1136 Create Account	net.exe	CMD: AND Contains “ user ”, “/ad”	1, 15, 11
	net1.exe	CMD: AND Contains “ user ”, “/ad” SPN: NOT Is “net.exe”	1, 15, 11
	–	CMD: AND Contains “New-LocalUser”, “-Name”	PE
T1038 DLL Search Order Hijacking	–	TOP: Contains “c:\temp”	7
T1158 Hidden Files and Directories	attrib.exe	CMD: OR Contains “+s”, “.bat”, “+h”	1, 15, 11
T1179 Hooking	mavinject.exe	CMD: Contains “/injectrunning”	1, 11, 15, 7, 3
T1062 Hypervisor	–	CMD: Contains “Hyper-V” CMD: OR Contains “Install-WindowsFeature”, “Get-WindowsFeature”	PE
T1037 Logon Scripts	reg.exe	CMD: AND Contains “hkcu\environment”, “userinitmprlogonscript”	1, 12, 13
T1031 Modify Existing Service	sc.exe	CMD: OR Contains “config”, “binPath=”	1, 15, 11
T1128 Netsh Helper DLL	netsh.exe	CMD: AND Contains “add helper”, “.dll”	1, 15, 11
	netsh.exe	TOP: Contains “\SOFTWARE\Microsoft\NetSh”	12, 13
T1050 New Service	sc.exe	SPN: cmd.exe CMD: OR Contains “create”, “start”, “stop”, “delete”	1, 15, 11, 7
	–	CMD: OR Contains “New-Service”, “Start-Service”, “Stop-Service”, (“Get-WmiObject Win32_Service” AND “.Delete()”)	PE
T1137 Office Application Startup	–	SPN: OR Contains “winword.exe”, “powerpnt.exe”, “excel.exe” PPP: OR Contains “fastprox.dll”, “wbemcomn.dll”, “wbemdisp.dll”, “wmiutils.dll”	7
	rundll32.exe	SPN: svchost.exe CMD: OR Contains “c08afd90-f2a1-11d1-8455-00a0c91f3880”, “9BA05972-F6A8-11CF-A442-00A0C90A8F39”	1, 15, 11, 12, 13
T1013 Port Monitors	–	TOP: Contains “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Monitors”	12, 13

Кінець таблиці 3.3

Техніка	TPN	Інші змінні	SEID / PE
T1060 Registry Run Keys / Startup Folder	reg.exe	CMD: OR Contains “SOFTWARE\Microsoft\Windows\CurrentVersion\Run”,“SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx”	1, 12, 13
T1180 Screensaver	reg.exe	CMD: Contains “\control panel\desktop” CMD: OR Contains “ScreenSaveActive”, “ScreenSaverTimeout”, “ScreenSaverIsSecure”	12, 13
T1101 Security Support Provider	–	CMD: Contains “not-a-ssp”	1, 15
	–	CMD: AND Contains “-ExpandProperty”, “Security Packages”	1, 15
	–	CMD: AND Contains “Get-ItemProperty HKLM:\System\CurrentControlSet\Control\Lsa”, “Security Packages”	1, 15
T1058 Service Registry Permissions Weakness	–	TOP: Contains “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services” TOP: OR Contains “FailureActions”, “FailureCommand”	12, 13
T1209 Time Providers	reg.exe	CMD: Contains “\SYSTEM\CurrentControlSet\services\W32Time\TimeProviders”	12, 13
T1084 WMI Event Subscription	–	CMD: OR Contains “New-CimInstance”, “__EventFilter”, “__FilterToConsumerBinding”, “CommandLineEventConsumer”, “Remove-WmiObject”, (“__InstanceModificationEvent” AND Contains “Query)	PE
T1004 Winlogon Helper DLL	powershell.exe/reg.exe	AND (CMD: Contains “Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify” OR TOP: Contains “Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify”) TON: OR Contains “logon”, “cmd.exe”	1, 12, 13
	powershell.exe/reg.exe	AND (CMD: Contains “software\microsoft\windows nt\currentversion\winlogon\shell” OR TOP: Contains “software\microsoft\windows nt\currentversion\winlogon\shell”) TON: OR Contains “shell”, “cmd.exe”	1, 12, 13
	powershell.exe/reg.exe	AND (CMD: Contains “software\microsoft\windows nt\currentversion\winlogon\shell” OR TOP: Contains “software\microsoft\windows nt\currentversion\winlogon\shell”) TON: OR Contains “userinit”, “cmd.exe”	1, 12, 13

Тактика Підвищення привілеїв (Privilege Escalation) налічує 21 техніку, із яких 5 не були покриті ніякими тестами. 16 технік мають покриття із тестів та правил на спрацювання. Atomic Red Team забезпечила 11 тестів для технік, 2 із

них без отриманих подій, частина технік була доповнена тестами на основі відомих індикаторів атак, інші ж 5 тестів були повністю покриті власними тестами та опрацьовані для отримання індикаторів. Перелік індикаторів технік тактики Підвищення привілеїв описано у Таблиці 3.4, яка подана нижче.

Таблиця 3.4 – Тактика Підвищення привілеїв (Privilege Escalation) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1088 Bypass User Account Control	powershell.exe/reg.exe	CMD: Contains “mscfile\shell\open\command” OR TOP: Contains “mscfile\shell\open\command”	1, 12, 13
	powershell.exe/reg.exe	(CMD: Contains “ms-settings\shell\open\command” OR TOP: Contains “ms-settings\shell\open\command”) AND (CMD: Contains “m fodhelper.exe” OR TOP: Contains “fodhelper.exe”)	1, 12, 13
T1055 Process Injection	–	CMD: Contains “etc/ld.so.preload”	1, 15, 11
	–	CMD: OR Contains “-ProcessID”, “Invoke-Dll”, “-Dll”	1, 15, 11
T1178 SID-History Injection	–	CMD: OR Contains “sIDHistory”, “Get-ADUser”	PE

У тактиці «Обхід захисту» (Defense Evasion) описуються техніки, за допомогою яких зловмисник може приховати шкідливу активність і запобігти своє виявлення засобами захисту. Різні варіації технік з інших тактик атаки, які допомагають подолати специфічні засоби захисту і превентивні заходи, включені в техніки обходу захисту. У свою чергу, техніки обходу захисту застосовуються на всіх фазах атаки.

Всього дана тактика містить 57 технік, при чому частина технік вже зустрічалася раніше, тому у Таблиці 3.5 подано ті техніки, які зустрілися вперше. 14 технік залишилися без покриття тестами, із них 4 залишають сліди тільки у подіях мережевих засобів. Atomic Red Team забезпечила 31 тест для технік, 2 із них без отриманих подій, частина технік була доповнена тестами на основі відомих індикаторів атак, інші ж 12 тестів були повністю покриті власними тестами та опрацьовані для отримання індикаторів. Також 12 технік від Atomic

Red Team були доповнені новими індикаторами, яких не було виявлено шляхом проведення їх тестів.

У Таблиці 3.5 подані індикатори для виявлення застосування технік у розрізі даної тактики «Обхід захисту».

Таблиця 3.5 – Тактика Обхід захисту (Defense Evasion) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1500 Compile After Delivery	csc.exe	CMD: OR Contains “.exe”, “.dll” AND CMD Contains “.cs”	1, 15, 11
T1140 Deobfuscate/Decode Files or Information	–	CMD: OR Contains “ copy ”, “certutil.exe”	1, 15, 11
	certutil.exe	CMD: OR Contains “-encode”, “-decode”	1, 15, 11
T1089 Disabling Security Tools	appcmd.exe	CMD: OR Contains “set config”, “/section:httplogging”, “/dontLog:true”	1, 15, 11
	fltmc.exe	CMD: OR Contains “unload”, “sysmon”	1, 15, 11
	–	CMD: Contains “ Set-MpPreference -DisableRealtimeMonitoring $true”	PE
	sc.exe	CMD: OR Contains “stop WinDefend”, “config WinDefend start= disabled”	1, 15, 11
	–	Value: Contains “1” TOP: OR Contains “ SOFTWARE\Policies\Microsoft\Windows Defender”, “DisableAntiSpyware”	12, 13
T1107 File Deletion	vssadmin.exe /wbadmin.exe/wmic.exe	CMD: Contains “delete”	1, 15, 11
	–	CMD: AND Contains “-path”, “Remove-Item”	PE
T1222 File Permissions Modification	takeown.exe/ cacls.exe/icacls.exe	–	1, 15, 11
	attrib.exe	CMD: Contains “-r”	1, 15, 11
T1484 Group Policy Modification	–	CMD: Contains “New-GPOImmediateTask”	PE
	–	CMD: Contains “MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf”	1, 15, 11
T1070 Indicator Removal on Host	wevtutil.exe	CMD: Contains “delete” AND CMD: Contains “cl”	1, 15, 11
	fsutil.exe	CMD: AND Contains “delete” , “usn”, “deletejournal”	1, 15, 11
T1202 Indirect Command Execution	forfiles.exe	–	1, 15, 11
	pcalua.exe	CMD: Contains “-a”	1, 15, 11

Кінець таблиці 3.5

Техніка	TPN	Інші змінні	SEID / PE
T1112 Modify Registry	reg.exe	CMD: OR Contains “SecurityHealth”, “HideFileExt”, “Software\Microsoft\Internet Explorer”, “CurrentContorlSet\Control\WMI\Security”, “Windows\CurrentVersion\Policies\System”	12, 13
T1096 NTFS File Attributes	–	FN: Contains “:” FN: OR Contains “.exe”, “.dll”	1, 15, 11
T1126 Network Share Connection Removal	–	CMD: OR Contains “net use”, “net share”	1, 15, 11, 3
T1027 Obfuscated Files or Information	–	SPN: OR Contains “cmd.exe”, “powershell.exe” CMD: OR Contains “^&”, “^^”, OR Matches “.*(^.^.^).*”, “.*(^..\\().*”	1, 15, 11
T1014 Rootkit	–	CMD: AND Contains “.sys”, “puppetstrings”	1, 15, 11
T1045 Software Packing	upx.exe	OR SPN: upx.exe	1, 15
T1221 Template Injection	cmd.exe/powershell.exe/explorer.exe	SPN: OR Contains “word.exe”, “powerpnt.exe”, “excel.exe”	1, 15
T1099 Timestomp	–	CMD: OR Contains “LastAccessTime”, “LastWriteTime”, “CreationTime”	1, 15, PE
T1497 Virtualization/ Sandbox Evasion	–	CMD: OR Contains “Win32_processor”, “Get-WmiObject”, “NumberOfCores”	PE

Під час тактики Доступ до облікових даних (Credential Access), отримавши облікові дані, зловмисник отримує доступ або навіть контроль над системою, доменом або службовими (технологічними) обліковими записами. Противник, ймовірно, буде намагатися дістати легітимні облікові дані користувача і адміністративних облікових записів, щоб ідентифікуватися в системі і отримати всі дозволи захопленого облікового запису, тим самим ускладнюючи процес виявлення його зловмисної активності. Противник також може створювати облікові записи з метою їх подальшого використання у цільовому середовищі.

Всього існує 16 технік, із яких 4 техніки без покриття тестами. Було використано 11 вже готових тестів для технік, розроблено для однієї техніки

тести, а також допрацьовано існуючі техніки для покращення ефективності знаходження слідів потенційно зловмисної активності із обліковими записами.

У Таблиці 3.6 подані індикатори для виявлення застосування технік у розрізі даної тактики.

Таблиця 3.6 – Тактика Доступ до облікових даних (Credential Access) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1110 Brute Force	–	CMD: AND Contains “ in ”, “net use”, “for /f”, “/user:”	1, 15, 11
T1003 Credential Dumping	–	CMD: OR Contains “gsecdump”, “wce.exe”, “procdump.exe”, “ntdsutil”, “vssadmin create shadow”	1, 15, 11, 3
	–	CMD: AND Contains “Invoke-Mimikatz”, “-DumpCreds”	PE
	–	CMD: OR Contains “reg save HKLM\sam”, “reg save HKLM\system” , “reg save HKLM\security”	12, 13
T1081 Credentials in Files	–	CMD: Contains “Invoke-mimikittenz”	PE
	–	CMD: AND Contains “select-string”, “-Pattern”	PE
	–	CMD: AND Contains “findstr /si”, “pass”	PE
T1214 Credentials in Registry	reg.exe	CMD: AND Contains “password”, “query”, “ /f”	1, 15, 11, 12, 13
T1056 Input Capture	–	CMD: AND Contains “Get-Keystrokes”, “-LogPath”	PE
T1141 Input Prompt	–	CMD: AND Contains “GetNetworkCredential().Password”, “UI.PromptForCredential”	PE
T1171 LLMNR/NBT-NS Poisoning and Relay	–	TOP: AND Contains “HKLM\Software\Policies\Microsoft\Windows NT\DNSClient”, “EnableMulticast”	12, 13
T1040 Network Sniffing	tshark.exe/dumpcap.exe	–	1, 15, 11
T1174 Password Filter DLL	–	TOP: AND Contains “SYSTEM\CurrentControlSet\Control\Lsa\”, “Notification Packages”	12, 13
T1145 Private Keys	–	CMD: OR Contains “cert.key”, “.key”	1, 15, 11

Отримавши в результаті первинної компрометації доступ в систему, противник розгортає діяльність по вивченню цільової системи та відслідковує можливості, які у нього з’явилися, і дізнається, чи достатньо поточних прав для

досягнення тактичної або кінцевої мети. Цей етап атаки називається «Виявлення» (Discovery).

Операційні системи мають безліч вбудованих інструментів, за допомогою яких противник може здійснювати дослідження внутрішнього периметра цільової мережі після її компрометації. У Windows для збору інформації можуть використовуватися інструменти прямої взаємодії з Windows API, функціонал WMI і PowerShell. Зловмисник застосовує методи виявлення під час вивчення середовища, тому виявлення подібної активності слід розглядати як ту частину атаки, за якою підуть спроби просування противника по мережі.

Усього налічується 22 техніки у межах даної тактики і усі ці техніки були покриті тестами. Із них 18 технік були розроблені від Atomic Red Team та частково доповнені новими індикаторами, а також інші 4 техніки було розроблено самостійно. У Таблиці 3.7 подані індикатори для виявлення застосування технік у розрізі «Виявлення».

Таблиця 3.7 – Тактика Виявлення (Discovery) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1087 Account Discovery	net1.exe/net.exe/query.exe	CMD: OR Contains “ user ”, “localgroup”	1, 15, 11
	–	CMD: Contains “query user”	PE
	–	CMD: Contains “cmdkey.exe”	PE
	–	(CMD: OR Contains “net ”, “get-“, “get”) AND (CMD: OR Contains “user ”, “localgroupmemebers”, “localgroup ”, “localgroupuser”)	PE
T1010 Application Window Discovery	csc.exe/cvtres.exe	CMD: Contains “-out:”	1, 15, 11
T1217 Browser Bookmark Discovery	–	CMD: OR Contains “AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\AC\MicrosoftEdge\User\Default\DataStore\Data ouser1\120712-0049\Favorites”, “AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\AC\MicrosoftEdge\User\Default\DataStore\Data ouser1\120712-0049\DBStore”, “AppData\Local\Google\Chrome\User Data\Default”, “AppData\Roaming\Mozilla\Firefox\Profiles\v8kf7cxv.default-release”	1, 15, 11

Продовження таблиці 3.7

Техніка	TPN	Інші змінні	SEID / PE
T1482 Domain Trust Discovery	nltest.exe	CMD: Contains “/domain_trusts”	1, 15, 11
	dsquery.exe	CMD: Contains “/domain_trusts”	1, 15, 11
T1083 File and Directory Discovery	–	CMD: Contains “-recurse” AND CMD: OR Contains “ls ”, “get-childitem”, “gci”	PE
	tree.com	–	1, 15, 11
T1046 Network Service Scanning	telnet.exe/Contains“nmap”	–	1, 15, 11, 3
	–	CMD: Contains “Get-Service”	PE
T1135 Network Share Discovery	net.exe	CMD: Contains “ view ”	1, 15, 11, 3
T1201 Password Policy Discovery	net1.exe/net.exe	CMD: Contains “accounts”	1, 15
T1120 Peripheral Device Discovery	reg.exe	CMD: Contains “query” AND CMD: OR Contains “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\”, “HKLM\SYSTEM\CurrentControlSet\Enum\”	12, 13
T1069 Permission Groups Discovery	net1.exe/net.exe	CMD: OR Contains “group /domain”, “localgroup”	1, 15
T1057 Process Discovery	tasklist.exe	–	1, 15
T1012 Query Registry	reg.exe	CMD: Contains “query”	1, 15
T1018 Remote System Discovery	net.exe	CMD: OR Contains “view”, “view /domain”	1, 15, 3
	ping.exe	CMD: OR Contains “-n”, “-w”	1, 15, 3
	arp.exe	CMD: Contains “-a”	1, 15, 3
T1063 Security Software Discovery	fltmc.exe	–	1, 15
	netsh.exe	–	1, 15
	findstr.exe	CMD: OR Contains “virus”, “cb”, “defender”, “cylance”	1, 15
T1082 System Information Discovery	systeminfo.exe	–	1, 15
	reg.exe	CMD: AND Contains “query”, “system\currentcontrolset\services\disk\enum”	1, 15
T1016 System Network Configuration Discovery	netsh.exe	CMD: Contains “interface show”	1, 15
	net.exe	CMD: Contains “config”	1, 15
	nbtstat.exe	CMD: Contains “-n”	1, 15
	arp.exe	CMD: Contains “-a”	1, 15
	ipconfig.exe	CMD: Contains “/all”	1, 15

Кінець таблиці 3.7

Техніка	TPN	Інші змінні	SEID / PE
T1049 System Network Connections Discovery	net1.exe/net.exe	CMD: Contains “sessions”	1, 15
	net1.exe/net.exe	CMD: Contains “use” AND NOT Contains “user”	1, 15
T1033 System Owner/User Discovery	–	CMD: OR Contains “whoami”, “quser /SERVER:”, “quser”, “wmic useraccount get”, “qwinsta.exe”	1, 15
T1007 System Service Discovery	sc.exe	CMD: OR Contains “start”, “query state”, “query”, “stop”	1, 15
	wmic.exe	CMD: AND Contains “service”, “like”, “where”	1, 15
T1124 System Time Discovery	w32tm.exe	CMD: Contains “/tz”	1, 15
	net1.exe/net.exe	CMD: Contains “time”	1, 15

Тактика Переміщення в мережі (Lateral Movement) включає методи отримання противником доступу і контролю над віддаленими системами, підключеними до цільової мережі, а також для запуску шкідливих інструментів на віддалених системах. Переміщення в мережі дозволяє зловмисникові отримувати інформацію з віддалених систем без використання додаткових інструментів. Всього описано 17 можливих технік, із них 7 було не покрито тестами. Atomic Red Team створила тести для 6 технік, 1 техніка не залишила слідів у подіях, 4 із цих технік були доповнені новими індикаторами, також додатково 2 теста був розроблений самостійно. У Таблиці 3.8 подані індикатори для виявлення застосування технік у розрізі даної тактики.

Таблиця 3.8 – Тактика Переміщення в мережі (Lateral Movement) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1037 Logon Scripts	reg.exe	CMD: AND Contains “hkcu\environment”, “userinitmprlogonscript”	1, 12, 13
T1076 Remote Desktop Protocol	–	CMD: AND Contains “tscon”, “rdp-tcp#”	1, 15, 3
T1105 Remote File Copy	certutil.exe	CMD: Contains “-urlcache -split -f”	1,15, 11, 3
	–	CMD: AND Contains “certutil”, “verifyctl -split -f”	PE
T1021 Remote Services	–	FP: OR Contains “putty”, “vncviewer.exe” Time is before and after work hours	1, 15, 3
T1077 Windows Admin Shares	–	CMD: AND Contains “admin”, “net use”	1, 15
	–	CMD: AND Contains “New-PSDrive”, “filesystem -root”	PE

Тактика Збір даних (Collection) та її техніки збору даних в скомпрометованому середовищі включають способи ідентифікації, локалізації і безпосереднього збору цільової інформації (наприклад, конфіденційних файлів) з метою її підготовки до подальшої ексфільтрації. Опис методів збору інформації також охоплює опис місць зберігання інформації в системах або мережах, в яких противники можуть здійснювати її пошук і збір.

Індикаторами реалізації більшості представлених в ATT&CK технік збору даних є процеси, що використовують API, WMI, PowerShell, Cmd для захоплення цільової інформації з пристроїв введення / виводу або множинного відкриття файлів на читання з подальшим копіюванням отриманих даних в певне місце в файловій системі або мережі . Інформація в ході збору даних може шифруватися і об'єднуватися в архівні файли.

Усього налічується 13 технік, із них тільки 6 були покриті тестами та правилами на спрацювання.

У Таблиці 3.9 подані індикатори для виявлення застосування технік у розрізі збору даних.

Таблиця 3.9 – Тактика Збір даних (Collection) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1123 Audio Capture	explorer.exe	CMD: AND Contains “.wma”, “shell:appsfolder\microsoft.windowssoundrecorder”	1, 15, 11
	powershell.exe	CMD: Contains “-command windowsaudiodevice-powershell-cmdlet”	1, 15, 11
T1119 Automated Collection	–	CMD: AND Contains “for /R”, “%f in”, “do copy”	1, 15, 11
	–	CMD: AND Contains “|”, “findstr /e”, “dir c: /b /s”	1, 15, 11
	–	CMD: AND Contains “Get-ChildItem -Recurse -Include”, “Copy-Item”	PE
T1115 Clipboard Data	clip.exe	PPP: OR Contains “cmd.exe”, “powershell.exe”	1, 15, 11

Тактика Командування і управління (Command and Control) включає техніки, за допомогою яких противник зв’язується із системами, підключеними до цільової мережі і які знаходяться під його керуванням. Залежно від конфігурації

систем і топології цільової мережі відомо безліч способів організації прихованого каналу C2.

Всього налічується 21 техніка, але покриття правилами було застосоване тільки до 4 із них, із решти не покритих тестами 8 технік містяться тільки у подіях мережевих засобів.

3 техніки із тестами були взяті у Atomic Red Team, 1 тест для техніки опрацьовано власноруч. У Таблиці 3.10 подані індикатори для виявлення застосування технік у розрізі даної тактики.

Таблиця 3.10 – Тактика Командування і управління (Command and Control) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1219 Remote Access Tools	–	FP: OR Contains “TeamViewer.exe”, “Ammyy_Service.exe”, “GoToAssist.exe”, “LogMeIn.exe”, “AmmyyAdmin”	1, 15
T1071 Standard Application Layer Protocol	–	CMD: AND Contains “Get-Random -Minimum”, “-type”, “Resolve-DnsName”, “-QuickTimeout”	PE
	–	CMD: AND Contains “-Domain”, “-QueryType”, “-C2Interval”, “-C2Jitter”, “-RunTime”	PE
	–	CMD: AND Contains “-Domain”, “-QueryType”, “-Subdomain”	PE
	–	CMD: AND Contains“Invoke-WebRequest”, “-UserAgent”	PE
T1065 Uncommonly Used Port	–	CMD: AND Contains “-ComputerName”, “-port” AND CMD: NOT Contains 25, 80, 443, 20, 21, 23, 143, 3389, 22, 53, 67, 68, 110	PE

Тактика Просочення даних (Exfiltration) або ж Вилучення даних описує техніки передачі даних, що застосовуються зловмисниками або шкідливим ПЗ для вилучення, крадіжки, витоку цільової інформації із скомпрометованої системи для досягнення поставленої цілі.

Техніки, які застосовуються у цій тактиці, достатньо складні та не завжди підлягають універсалізації, тому для індикаторів було обрано ряд команд, які потенційно можуть бути застосованими для різних ситуацій і є максимально універсальними.

Усього було виявлено 9 технік, із яких 3 були покриті правилами на спрацювання у SIEM системі і протестовані у тестовому середовищі для збору

подій та отримання індикаторів протестованих технік. У Таблиці 3.11 подані індикатори, за якими здійснювалося виявлення застосування технік у розрізі даної тактики. Із таблицею можна ознайомитися нижче.

Таблиця 3.11 – Тактика Просочення даних (Exfiltration) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1002 Data Compressed	rar.exe	CMD: Contains “ r ”	1, 11, 15
	–	CMD: Contains “Compress-Archive -DestinationPath”	PE
T1022 Data Encrypted	rar.exe	CMD: Contains “-hp”	1, 11, 15
	winzip64.exe /winzip32.exe	CMD: Contains “-s”	1, 11, 15
	7z.exe	CMD: Contains “-p”	1, 11, 15
T1048 Exfiltration Over Alternative Protocol	–	CMD: Contains “System.Net.Networkinformation.ping”	PE
	–	CMD: AND Contains “-Encoding Byte -ReadCount”, “Get-Content -Path”, “.Send”	PE

Тактика «Вплив» (Impact) – остання стадія у розрізі життєвого циклу APT атаки – складається із технік, які супротивники використовують для порушення доступності або порушення цілісності системи, маніпулюючи бізнес- та операційними процесами. Методи, що застосовуються для впливу, можуть включати знищення чи підробку даних.

Усього існує 14 технік, із яких 4 техніки було опрацьовано та отримано для них події із індикаторами, що можуть говорити про потенційно зловмисну активність, решта 6 технік залишають сліди тільки у подіях мережевих засобів і не можуть бути викриті через sysmon і powershell події, тому дані техніки залишилися без покриття індикаторами та правилами на спрацювання у SIEM.

У таблиці 3.12 подані індикатори для виявлення застосування технік у розрізі даної тактики Вплив, із якими можна ознайомитися нижче. Індикатори було максимально універсалізовано.

Таблиця 3.12 – Тактика Вплив (Impact) – індикатори компрометації для технік

Техніка	TPN	Інші змінні	SEID / PE
T1485 Data Destruction	–	CMD: Contains “sdelete.exe”	1, 11, 15
T1490 Inhibit System Recovery	–	CMD: OR Contains “wmic.exe shadowcopy delete”, “vssadmin.exe delete shadows /all /quiet”, “rdp-tcp#”, “wbadmin.exe delete catalog -quiet”,	1, 11, 15
T1488 Disk Content Wipe_registry event	reg.exe	TOP: OR Contains “System\CurrentControlSet\Control\SystemBootDevice”, “System\CurrentControlSet\Control\FirmwareBootDevice”	12, 13
T1489 Service Stop	sc.exe	CMD: Contains “stop”	1, 15
	net.exe	CMD: Contains “stop”	1, 15
	taskkill.exe	CMD: Contains “/f /im”	1, 15

3.4 Програмна реалізація етапів розробленого методу

Для реалізації таких етапів методу, як: побудова взаємозв’язків між процесами за допомогою алгоритму нечіткого пошуку, побудова послідовності застосування технік MITRE та співставлення із життєвим циклом APT атаки, визначення хронології подій APT атаки – був розроблений спеціальний застосунок мовою програмування Java із використанням бібліотек XChart, Graphstream, Apache CSV для зчитування даних та їх подальшого графічного відображення. Початково на вхід до застосунку дається файл із подіями, файли з техніками та тактиками, техніками та їх ідентифікаторами. На виході формуються графіки із хронологією технік APT та список відповідних подій.

Всього були створені такі класи (Додаток А): Chart.java, Event.java, EventGraph.java, BuildAPT.java, IdTechniques.java, Levenstein.java, ParserCsv.java, Tactics.java, TechniqueChart.java, Techniques.java. Головним класом є BuildAPT.java, у якому відбувається запуск функцій для реалізації усіх алгоритмів відбору даних. На Рис.3.3 представлена UML діаграма класів розробленого застосунку.

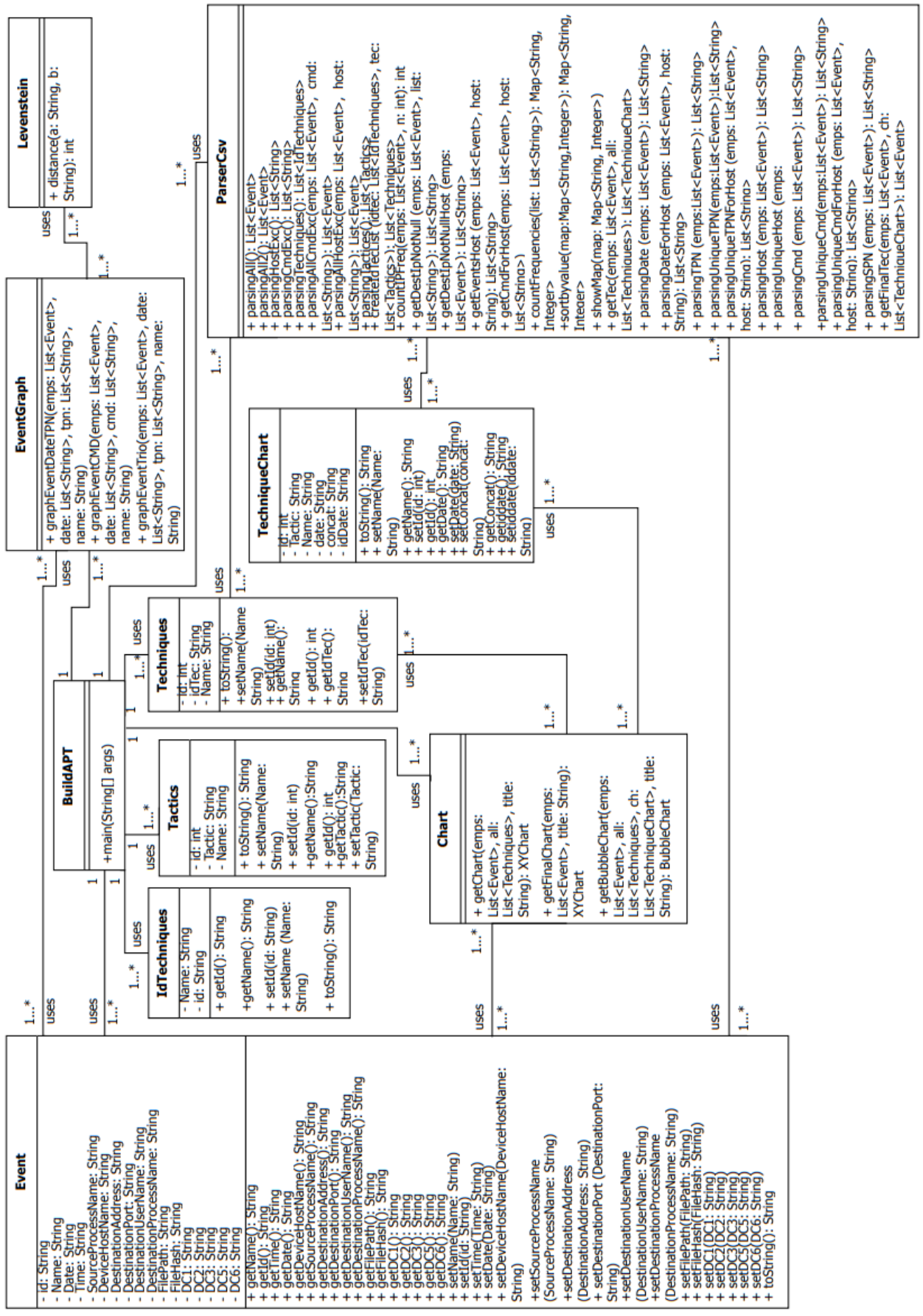

Рисунок 3.3 – Діаграма UML класів застосунку

Для коректного розбору даних за полями було реалізовані спеціальні типи даних. У класі Event.java описано тип даних для події, отриманої із файлу з кореляційними подіями від Sysmon та Powershell. Виділено основні характеристики події (типом даних для кожного із параметрів є String): унікальний ідентифікатор події (id), ім’я події (Name), дата (Date), час (Time), назва хосту (DeviceHostName), назва джерела процесу (SourceProcessName), адреса призначення (DestinationAddress), порт призначення (DestinationPort), ім'я цільового користувача (DestinationUserName), назва цільового процесу (DestinationProcessName), шлях до файлу (FilePath), хеш файлу (FileHash), командний рядок (DC1) або ж значення призначеного параметру у випадку події з реєстру, шлях до батьківського процесу (DC2) або ж повний шлях із назвою процесу чи сама назва, командний рядок батьківського процесу (DC3), тільки шлях до батьківського процесу (DC5), підпис бібліотеки (DC6). Також для кожного із вказаних полів було реалізовано по дві функції для задання параметру та для його отримання. Додатковою функцією для цього класу є виведення у рядок основних параметрів події *toString()*.

Наступний тип даних реалізований у класі IdTechniques.java, що потрібен для парсингу файлу зі списком усіх технік та їх унікальних ідентифікаторів, ключовими полями цього типу є ідентифікатор (id) та ім'я техніки (Name), доступні функції для задання параметру та для його отримання. Тип даних для полів – String.

Тип даних Techniques.java створений для подачі технік у розрізі ідентифікатору техніки (idTec), її імені (Name) та порядкового номеру стадії APT (id) – іншими словами, тактики у розрізі MITRE. Тип даних для параметрів – String. Функції для задання та отримання того чи іншого ключового параметру реалізовані за замовчуванням.

Tactics.java реалізований для збереження даних після парсингу даних про тактики MITRE. Основні параметри – назва тактики (String Tactic), її порядковий номер (int id), назва техніки (String Name), що застосовується у розрізі цієї

тактики. Також створені функції для задання вищеописаних параметрів та повернення їх значення.

TechniqueChart.java – спеціальний клас для опису типу даних, який призначений для побудови графіків у класі Chart.java. Основними параметрами даного класу є: порядковий номер стадії APT (id), дата (date), ідентифікатор техніки MITRE (Name), параметр, що поєднує дату та ідентифікатор техніки (concat), параметр, що об'єднує такі змінні, як ідентифікатор техніки, порядковий номер стадії APT та дату (iddate). Також реалізований ряд функцій для задання та повернення значень параметрів класу, функція для повернення ключових значень (id, Name, date).

Алгоритм Вагнера-Фішера було реалізовано як окремий клас Levenstein.java. У даному класі описана функція *distance (String a, String b)* для обрахунку відстані Левенштейна у рамках алгоритму Вагнера-Фішера. Обрахунок цієї відстані викликається у функціях класу EventGraph.java, які будуть описані далі нижче.

У класі ParserCsv.java налічується 30 функцій, призначених для парсингу подій та виділення певних параметрів зі списків. Дані функції та опис їх призначення подано у Таблиці 3.13.

Таблиця 3.13 – Функції класу ParserCsv

Функція та параметри на вхід	Призначення функції
List<Event> parsingAll ()	Парсинг подій із файлу формату .csv, в якому зберігаються усі події за техніками MITRE, за ключовими параметрами типу даних Event та додавання кожної події у список типу Event.
List<Event> parsingAll2 ()	Парсинг подій із файлу формату .csv, в якому зберігаються кінцеві події за техніками MITRE, за ключовими параметрами типу даних Event.
List<Event> parsingAllCmdExc (List<Event> emps, List<String> cmd)	Функція видалення подій із типовими для організації командних рядків із початкового списку та збереження у новому списку типу Event.
List<Event> parsingAllHostExc (List<Event> emps, List<String> host)	Функція видалення подій із хостами, на яких не було виявлено підозрілої діяльності, із початкового списку та збереження у новому списку типу Event.
List<String> parsingHostExc ()	Парсинг списку із переліком хостів, , на яких не було виявлено підозрілої діяльності.

Продовження таблиці 3.13

Функція та параметри на вхід	Призначення функції
List<String> parsingCmdExc ()	Парсинг списку із переліком командних рядків, які є нормальними у розрізі організації.
List<IdTechniques> parsingTechniques ()	Парсинг списку із переліком усіх технік MITRE та їх відповідними унікальними ідентифікаторами.
List<Tactics> parsingTactices ()	Парсинг списку, що описує матрицю MITRE, де вказано порядковий номер тактики у розрізі APT, назву тактики, назва техніки, що належить даній тактиці.
List<Techniques> createIdTecList (List<IdTechniques> idtec, List<Tactics> tec)	Функція для створення об'єднаного списку із порядкового номеру тактики, ідентифікатору та назви техніки, які належать тактиці.
int countIPFreq (List<Event> emps, int n)	Функція обрахунку кількості не порожніх адрес призначення у списку подій.
List<String> getDestIpNotNull (List<Event> emps, List<String> list)	Пошук та збереження не порожніх адрес призначення у список формату String.
List<String> getDestIpNotNullHost (List<Event> emps)	Пошук та збереження у списку хостів, із яких спостерігалися підключення до зовнішніх адрес.
List<Event> getEventsHost (List<Event> emps, String host)	Пошук усіх хостів зі списку подій.
void getCmdForHost(List<Event> emps, List<String> host)	Виведення поєднання хост та командний рядок із події.
Map<String, Integer> countFrequencies(List<String> list)	Підрахунок кількості будь-якого параметру зі String списку.
Map<String, Integer> sortbyvalue(Map<String, Integer> map)	Сортування частоти виникнення певного параметру за спаданням.
void showMap(Map<String, Integer> map)	Виведення поєднання ключ та значення.
List<String> parsingDate (List<Event> emps)	Пошук та збереження унікальних дат у списку подій.
List<String> parsingDateForHost (List<Event> emps, String host)	Пошук та збереження унікальних дат у списку подій для певного хосту.
List<String> parsingTPN (List<Event> emps)	Пошук та збереження усіх цільових процесів зі списку подій.
List<String> parsingUniqueTPN (List<Event> emps)	Пошук та збереження унікальних цільових процесів зі списку подій.
List<String> parsingUniqueTPNForHost (List<Event> emps, String host)	Пошук та збереження унікальних цільових процесів зі списку подій для певного хосту.
List<String> parsingHost (List<Event> emps)	Пошук та збереження усіх хостів зі списку подій.

Кінець таблиці 3.13

Функція та параметри на вхід	Призначення функції
List<String> parsingUniqueHost (List<Event> emps)	Пошук та збереження унікальних хостів зі списку подій.
List<String> parsingCmd (List<Event> emps)	Пошук та збереження усіх командних рядків зі списку подій.
List<String> parsingUniqueCmd (List<Event> emps)	Пошук та збереження унікальних командних рядків зі списку подій.
List<String> parsingUniqueCmdForHost (List<Event> emps, String host)	Пошук та збереження унікальних командних рядків зі списку подій для певного хосту.
List<String> parsingSPN (List<Event> emps)	Пошук та збереження усіх процесів із джерела зі списку подій.
List<Event> getFinalTec(List<Event> emps, List<TechniqueChart> ch)	Виведення та збереження кінцевого списку подій потенційної APT.
List<TechniqueChart> getTec(List<Event> emps,List<Techniques> all)	Компонування списку типу TechniqueChart для подальшого використання при побудові графіку.

Саме у цьому класі реалізовані функції, які відповідають за проведення підготовчих дій для отримання потрібних даних, що згодом будуть використанні в інших функціях.

Для графічного відображення взаємозв'язків події між процесами, датами, командними рядками і так далі було використано дві бібліотеки – Graphstream для побудови графів та XChart для створення графіків різних типів. Для парсингу подій із файлів формату .csv було використано бібліотеку CSV від Apache. Потреба у графічному відображенні подій та взаємозв’язків між ними пов’язана із тим, що при розгляді текстового відображення усіх подій аналітиком дуже важко встановити зв’язки між подіями, визначити легітимність процесів чи параметрів командного рядка. Але під час агрегації подій за певними полями та відображенні їх на часовому проміжку полегшує завдання розрізнення нормальної діяльності та зловмисної, а також визначення потенційних зловмисних утручань.

Клас Chart відповідає за такі етапи розробленого методу аналізу APT атак: побудова послідовності застосування технік MITRE та співставлення із життєвим циклом APT атаки, визначення хронології подій APT атаки.

Chart.java містить основні функції для створення відображення хронології подій із технік на основі різних масивів даних. Функція *XYChart getChart(List<Event> emps, List<Techniques> all, String title)* будує звичайний графік на основі списку із подій, із яких обирає використану техніку у розрізі дня та ставить її у співвідношення зі стадією APT. На вхід до функції передаються список подій, список технік з їх ідентифікаторами, назва майбутнього графіку. Функція *XYChart getFinalChart(List<Event> emps, String title*) по структурі наслідує попередню функцію та призначена для відображення кінцевої хронології потенційної APT атаки. Також існує ще одна функція *BubbleChart getBubbleChart(List<Event> emps, List<Techniques> all, List<TechniqueChart> ch, String title)* для створення особливого типу графіку. За вхідними типами даними ця функція ідентична до двох попередніх, але тип графіку *BubbleChart* визначає, що у графіку точка із позначенням використаної техніки на пересіченні дати та стадії APT буде описуватися не одиничним діаметром, а діаметром, що відповідає кількості використання цієї техніки в день (не залежно від хосту, просто в цілому). Кожна із цих функцій повертає графік та створює його графічне відображення.

Ще одним класом для графічного відображення взаємозв'язків між подіями з певними однаковими параметрами у розрізі днів, а також робочих станцій, є EventGraph.java. Цей клас реалізує етап методу аналізу APT атак – побудова взаємозв'язків між процесами за допомогою алгоритму нечіткого пошуку. Під час сукупного розгляду подій було зроблено висновок, що найбільш важливими параметрами для пов'язування подій є цільові процеси та значення командних рядків, так як саме ці поля є найбільш інформативними та можуть описати потенційно зловмисні дії у деталях. Інші ж поля зі змінної типу *Event* є допоміжними або ж такими, що доповнюють подію до визначених параметрів. Перша функція *void graphEventDateTPN(List<Event> emps, List<String> date, List<String> tpn, String name)* на вхід приймає список подій, список дат та цільових процесів, які обраховані на основі списку подій, а також назва для графу. Типи вершин для графу – дати та цільові процеси. Ребра графа будуються на

основі підрахунку відстані Левенштейна для дат, далі, якщо для певної дати існувала подія використання техніки з певним цільовим процесом, то дане ребро з'єднує відповідні вершини з датою та з цільовим процесом. Функція при її викликанні будує відповідний граф та відображає його у паралельному потоці.

Функція *void graphEventCMD(List<Event> emps, List<String> date, List<String> cmd, String name)* отримує на вхід список із подіями, відповідні до подій списки з датами та командними рядками, а також назву для графу. Даний граф містить вершини із датою та вершини із командними рядками, взаємозв'язки між вершинами будується на основі дати та командного рядка, який був присутній у події у розрізі даного дня. Для пов'язування командних рядків із датами використовується обчислення відстані Левенштейна для полів із події та записів зі списків дат та командних рядків. На виході будується граф із відображенням дат та використаних значень командних рядків для кожної із дат.

Найбільш комплексною функцією з точки зору побудови графу є *void graphEventTrio(List<Event> emps, List<String> date , List<String> tpn, String name).* Дана функція має три типи вершин: дата, цільовий процес, командний рядок. Для зрівняння ключових полів із події та елементів переданих списків з датами, процесами та командними рядками також використовується функція Левенштейна, яка обмежується певним значенням (у більшості випадків це 0), який визначає рівень схожості двох порівнюваних текстових значень. Побудова ребер графу відбувається у наступній послідовності: спочатку визначаються взаємозв'язки між датою та цільовими процесами, що були використані у відповідний день, далі відбувається пошук тих подій, де дата та цільовий процес має заданий рівень схожості із наявними вершинами та ребрами, та будує ребро від процесу до знайденого командного рядка у розрізі певної дати. На виході повертається граф із відображенням відповідних взаємозв'язків.

BuildAPT.java – головний клас застосунку, у якому створюються об'єкти та викликаються функції для аналізу даних та побудови хронології подій APT. Основний алгоритм дій у головному класі:

1. Створення об'єкту класу ParserCsv.

2. Парсинг подій із вказаного файлу за допомогою функції parsingAll(), збереження даних до списку типу Event.
3. Парсинг ключових полів в окремі списки типу String: дати, цільові процеси, робочі станції, командні рядки, процеси джерела, а також парсинг унікальних значень вищевказаних полів в інші окремі списки.
4. Парсинг файлів із файлів із техніками та тактиками, техніками та їх ідентифікаторами для створення одного загального списку типу Techniques.
5. Створення графіків та графів на основі початкового списку подій.
6. Аналіз графіків та графів аналітиком, виведення списку нормальних командних рядків, які відповідають типовій діяльності організації.
7. Парсинг файлу із виключеннями командних рядків, пошук таких командних рядків у початковому файлі для аналізу та вилучення відповідних подій, збереження результатів у новий список типу Event.
8. Повторення пунктів 3-6 на основі нового списку.
9. Після аналізу графічних відображень виведення списку робочих станцій (якщо такі наявні), на яких не була відображена небезпечна або потенційно небезпечна діяльність.
10. Парсинг файлу із виключеннями командних рядків та файлу із виключенням робочих станцій, пошук таких командних рядків та станцій у попередньому файлі подій для аналізу та вилучення відповідних подій, збереження результатів у новий список типу Event.
11. Повторення пунктів 3-6 та 9-10 на основі нового списку до тих пір, доки не залишиться таких подій, які можна було б назвати нормальними та така діяльність була підтверджена організацією нормальною.
12. Виклик функції getTec на основі останнього аналізованого файлу типу Event для отримання списку типу TechniqueChart із даними для побудови кінцевих графіків та графів.
13. Вивантаження кінцевих даних за допомогою функції getFinalTec на основі останнього аналізованого файлу та попереднього списку типу TechniqueChart.

Наприкінці, аналітиком буде отримана графічна схема хронології подій на графіку, а також послідовне відображення хронології APT із описом застосованих технік та їх призначення у розрізі атаки.

Висновки до розділу 3

У рамках третього розділу було розроблено метод аналізу APT атак на основі технік MITRE та із застосуванням алгоритму нечіткого пошуку. Для цього було проаналізовано матрицю MITRE Enterprise для Windows, а саме тактики та техніки, які використовуються в рамках тактики. Також було проаналізовано механізми реалізації технік, застосовано їх у тестовому середовищі для отримання подій від Sysmon та Powershell. На основі отриманих подій було виведено список індикаторів застосування технік для кожної із тактик та створено правила в ArcSight ESM для отримання кореляційних подій. Дані роботи були проведені як частина реалізації етапу методу – фільтрація подій.

Наступні етапи методу було реалізовано у програмному застосунку мовою Java. Основним призначенням застосунку є зчитування збережених подій із файлу та проведення ряду дій на основі цього файлу: пошук подій, класифікованих «нормальні/планові для організації», відсіювання таких подій, побудова хронології виконаних технік на усіх робочих станціях так само і на окремих, виведення взаємопов’язаних подій за датою, цільовим процесом, командним рядком, отримання унікальних значень командних рядків, цільових процесів, робочих станцій, процесів джерела, адрес призначення. Для цього у застосунку було реалізовано 10 класів та набір відповідних функцій, які виконують усі вищеописані дії та ряд інших допоміжних функцій. Кінцевим результатом роботи застосунку передбачено вивід кінцевої хронології подій потенційної APT та списку усіх подій, які були класифіковані як APT.

4 АНАЛІЗ РЕЗУЛЬТАТІВ

У четвертому розділі проведено аналіз отриманих результатів роботи розробленого методу аналізу APT через застосування його програмної реалізації до обраних вхідних даних. Було описано почергове проходження усіх етапів методу та дій, які були застосовані на кожному із них. Як результат роботи методу було подано опис виявленої потенційної APT у розрізі застосованих технік.

4.1 Вхідні дані для аналізу

Першим етапом при проведенні тестування роботи методу аналізу APT атак був вибір даних для аналізу. Під час огляду можливостей використання справжнього сету даних APT атаки виявилося, що таких даних у відкритому доступі немає, тестових даних, у яких були б події від Sysmon та Powershell теж знайти не вдалося. Тому було обрано альтернативний шлях вирішення цієї проблеми. На основі даних організації, для якої було проведено створення та налаштування правил на основі індикаторів технік MITRE для SIEM ArcSight, було проведено аналіз потенційної APT.

Особливістю даних є те, що частиною робочих станцій були тестові машини, на яких проводилося тестування технік MITRE та запуск потенційно зловмисних дій. Також проводилася фіксація дій і звичайних користувачів, на основі чого періодично формувалися спрацювання правил. Як мінімум, на основі тестових запусків технік було створено тестову APT, але внаслідок тривалого проведення експерименту її повний обсяг та опис за етапами життєвого циклу APT був достеменно невідомий. Тестові дані та спрацювання від інших користувачів відбувалися паралельно, тому явного вирізнення тестових даних у суцільному потоці подій не було, а також весь сет даних піддавався аналізу, без

вирізнення тих робочих станцій, на яких точно були здійсненні тестові запуски технік.

Всього період збору даних зайняв 6 місяців – від початку травня (для деяких робочих станцій від кінця квітня) до кінця жовтня поточного року. Дані були вивантажені у форматі .csv, після чого зазнали процесу знеособлення – перейменування назв робочих станцій, користувачів, видалення чи заміна внутрішніх IP адрес.

На момент початку проведення аналізу об’єм даних складав 10987 подій від Sysmon та Powershell, які підлягають умовам створених правил за індикаторами технік та були створені внаслідок спрацювання відповідних правил.

4.2 Проведення аналізу за розробленим методом

Першим етапом у методі аналізу APT є фільтрація даних на основі технік MITRE. Цей етап був реалізований шляхом створення правил для SIEM ArcSight, як і описувалося раніше. У попередньому пункті було вказано які саме дані були використані для проведення аналізу та як вони були отримані.

Ці дані є вхідними для наступного етапу – застосування методу нечіткого пошуку. Після отримання початкового набору даних було перевірено чи всі поля із подій співпадають зі змінними, в які вони будуть передаватися під час парсингу файлу у застосунку. Згідно з описаним у попередньому розділі алгоритмом проведення аналізу за допомогою розробленого застосунку було відтворено усі вказані етапи.

Спочатку дані були опрацьовані функцією parsingAll() та збережені до списку типу Event. Також на основі створеного списку було виділено списки типу String для збереження ключових полів: дати, цільові процеси, робочі станції, командні рядки, процеси джерела, а також створені списки для збереження унікальних значень цих полів.

Для вирізнення етапів APT(порядкові номери тактик) та технік, які можуть бути застосовані на кожному із етапів, було створена два файли на основі даних із офіційного сайту MITRE. У першому файлі міститься інформація про порядковий номер тактики, її назву, назву відповідної техніки, яка застосовується для цієї тактики. У другому файлі міститься список із унікальним ідентифікатором техніки та її назвою. Ці два списки призначені для створення списку типу Techniques із такими основними параметрами: порядковий номер тактики, ідентифікатор техніки та її назва.

Другий етап методу аналізу напряму реалізується шляхом створення графів класу EventGraph на основі початкового списку подій та допоміжних списків за ключовими параметрами події. У форматі тесту було створено граф для усієї організації на базі дати, цільового процесу та командного рядку. Але в результаті був отриманий переповнений інформацією граф, у якому важко відслідкувати частоту виникнення ключових полів та зробити висновок про їх легітимність. Тому було вирішено, що оптимальним шляхом вирішення цього питання є побудова графів у розрізі кожної із робочих станцій. Приклад графів для однієї із робочих станцій поданий на Рис.4.1 нижче.

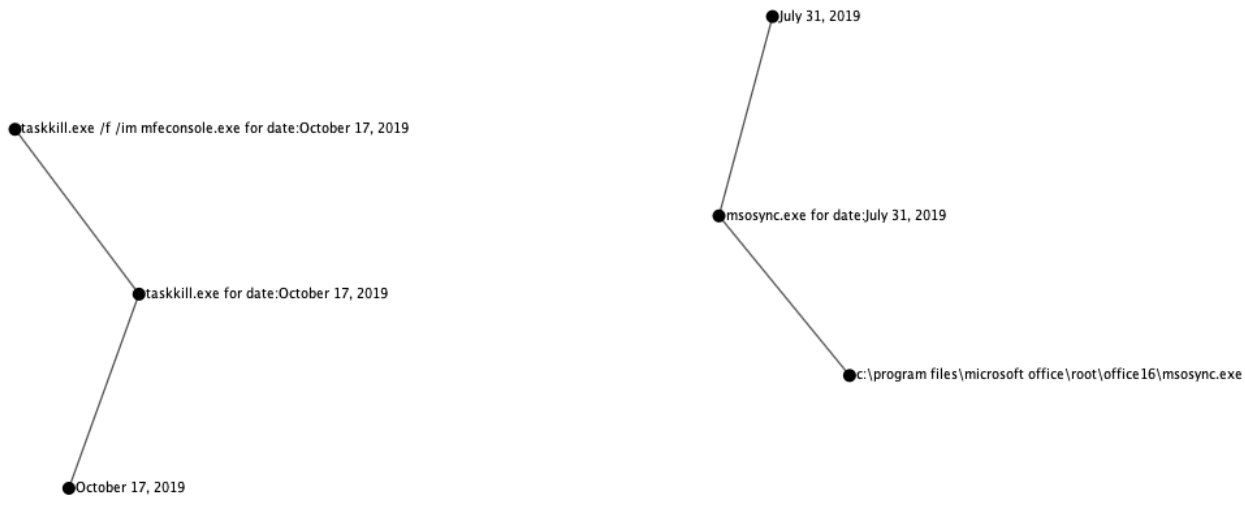

Рисунок 4.1 – Приклад графів для однієї робочої станції

У даних графах основною інформацією є пов’язування подій у граф із використання алгоритму Вагнера-Фішера через зрівняння відстані Левенштейна для дат із окремого списку дат та дат у основному списку подій, цільових

процесів у окремому списку та основному списку подій, а також для командних рядків у окремому списку рядків та основному списку подій. Функції, які забезпечують цей функціонал були описані вище у попередньому розділі. Такі графи були побудовані для кожного із хостів, наявних у списку робочих станцій, також були створені графи на основі пов’язування дат та цільових процесів, дат та командних рядків для відслідковування типових для організації хостів.

Із використанням функцій класу Chart було побудовано графіки послідовності застосування технік MITRE та співставлення із життєвим циклом APT атаки для кожної із робочих станцій та загальний вигляд послідовності для усіх станцій на основі початкового сету подій. Кожна точка на графіку – це техніка, яка була експлуатована у певний день хоча б на одному хості. На Рис.4.2 відображена початкова послідовність використання технік для усієї організації на початковому етапі.

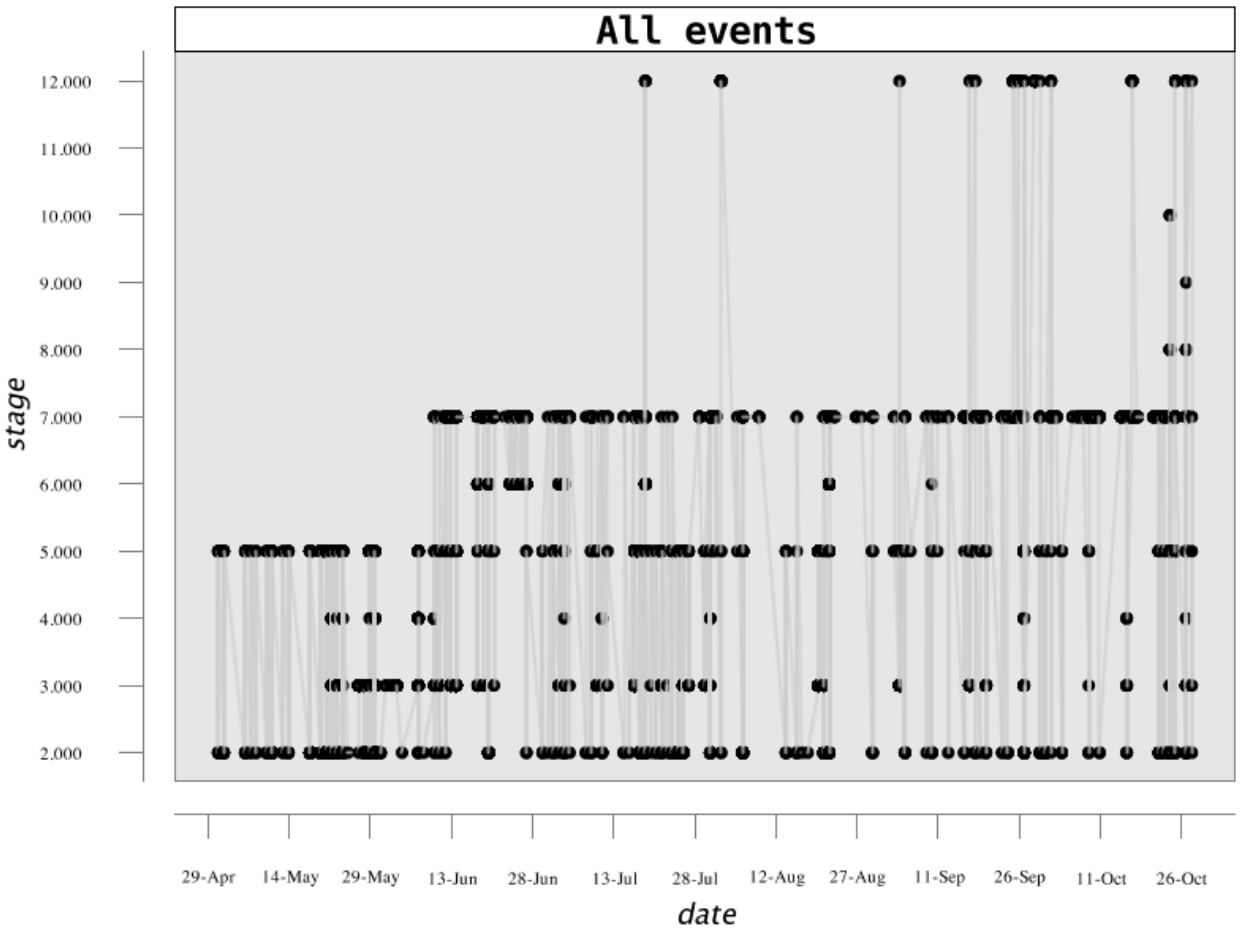

Рисунок 4.2 – Графік послідовності застосування технік на початковому етапі

Як бачимо у розрізі організації кількість застосованих технік протягом терміну у півроку є достатньо великою. Якщо в день було застосовано одну або більше ніж одну техніку на одному або декількох хостах, то на графіку ставилася точка на перетині дати та порядкового номеру тактики. Але є такі техніки, які можуть бути застосовані на декількох етапах APT, тому, як бачимо, для даних в кінці квітня – на початку травня спостерігається виникнення технік із 2 та 5 тактики. Як показав детальний аналіз, це виявилася одна і та ж сама техніка, тому цей факт був врахований для побудови графіку на основі кінцевого сету даних – у цьому випадку, вручну обиралося до якої тактики належить та чи інша техніка, якщо це ранній етап – то відповідь схилялася до тактики із меншим порядковим номером, а якщо етап був пізнім – то навпаки, до техніки із більшим порядковим номером.

Під час аналізу графів із командними рядками було зіставлено список тих командних рядків, які допускаються в межах організації, та визначені як легітимні.

Додатковим типом аналізу було проведення підрахунку кількості унікальних командних рядків для усієї організації, що дало результативну характеристику частоти застосування тих чи інших параметрів у командних рядках на різних робочих станціях. Такий тип пошуку також здійснювався і для інших ключових параметрів події (цільовий процес, процес джерела, адреси призначення) для кращого аналізу та пошуку потенційної APT.

Прикладом такого командного рядка може бути випадок, коли процес webexmta.exe ініціює запуск утиліти ipconfig.exe шляхом виклику через командний рядок команди ipconfig /all. Така діяльність цілком нормальна для цього процесу, тому дана пара командний рядок та батьківський процес були внесені як виключення та в подальшому були видалені зі списку подій для зменшення кількості фальш-позитивних спрацювань.

Іншим прикладом нормального командного рядка є діяльність VPN агенту sgpm.exe, що знаходиться у папці c:\program files (x86)\forcepoint\vpn client\sgpm.exe: «"c:\windows\system32\netsh.exe" interface ipv4 add route

"%IP%"/32 "%IP%" interface=3 store=active» та інші подібні команди. Цей командний рядок був помічений у техніці T1063 Security Software Discovery, яка відповідає тактиці з порядковим номером 7. Процес sgpm.exe був попередньо перевірений за хешем на відомих пісочницях virustotal, exchange.xforce.ibmcloud.com, hybrid-analysis, які підтвердили відсутність рівня ризику від цього процесу. Також в організації було проведено запит на легітимність даного VPN агенту. На одному із хостів видно частоту та періодичність виникнення цієї техніки, що показано на Рис.4.3.

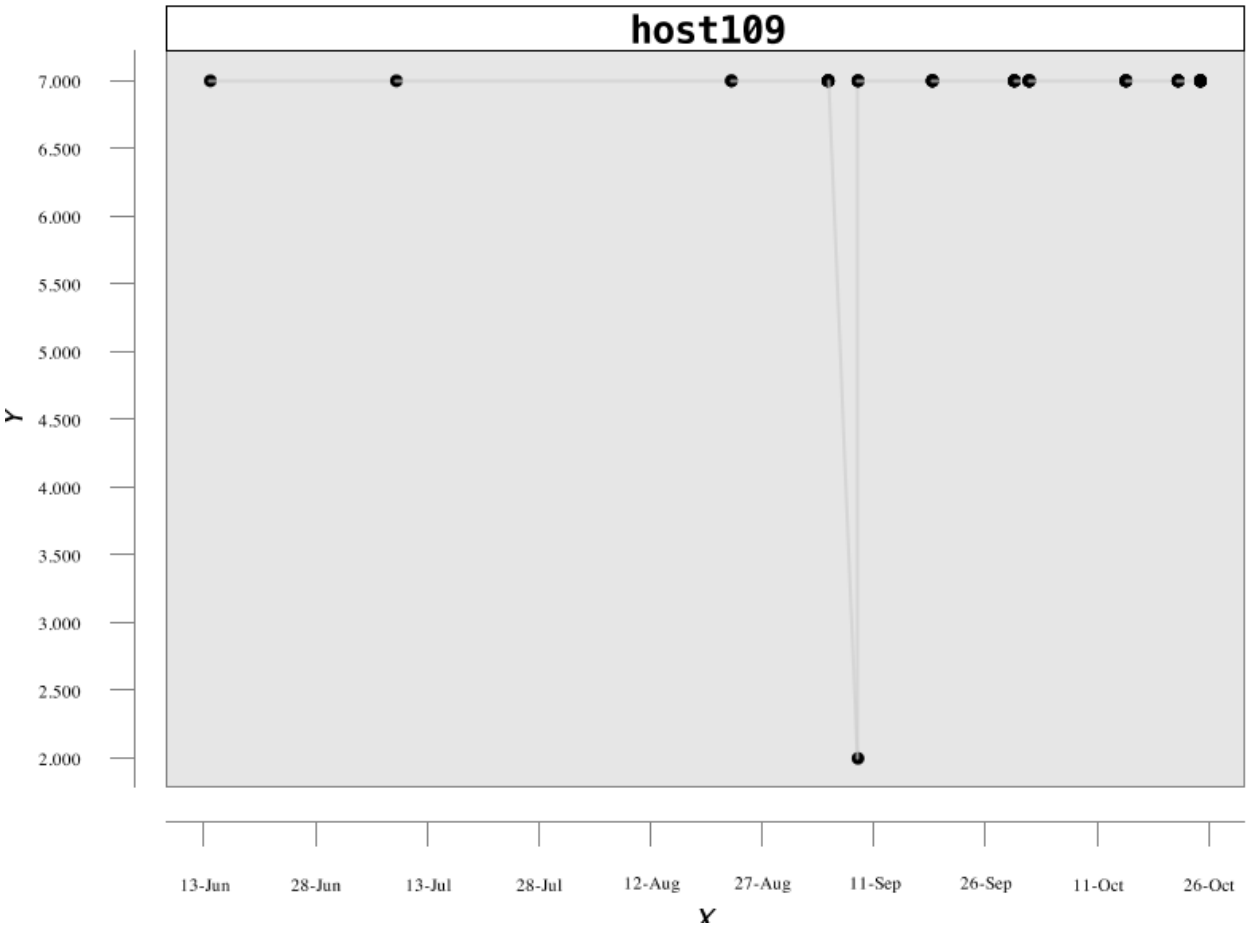

Рисунок 4.3 – Графік послідовності технік

Як видно з ілюстрації вище, згадана техніка використовувалася на цьому хості та інших також протягом певного проміжку часу на постійних засадах. Під час перевірки легітимності даного запуску на офіційному сайті постачальника даного продукту було виявлено, що така діяльність є нормальною. Тому цей командний рядок був доданий до виключень.

На основі подібних випадків був створений список із командними рядками, які є нормальними у розрізі усієї організації. Цей список був застосований до початкового списку подій, із якого були вилучені вказані командні рядки та був створений новий список типу Event. Після видалення нормальних командних рядків із початкового списку кількість подій із 10987 був зменшений до 7025.

Після цього були проведені повторні дії 3-6 із алгоритму дій для розробленого застосунку. Було побудовано графіки та графи для робочих станцій, які піддалися детальному огляду. Таким чином був проаналізований новостворений список та виділені цільові процеси та процеси джерела, які внеслися до нових списків виключень, також здійснилася перевірка хостів, чи наявні там такі техніки, які відбулися в рамках нормальної діяльності, а не потенційної APT атаки.

Наприклад, на Рис.4.1 зображені графи для одного із хостів, на якому підозрілої діяльності виявлено не було. Тобто з огляду на описані події у графах було проведено аналіз відповідних процесів та командних рядків, що підтвердили нормальну діяльність для цієї робочої станції. Тобто цей хост був занесений до списку виключень.

На основі цих списків виключень знову була проведена процедура відсіювання нормальних подій зі списку. Кількість подій порівняно із попереднім списком у 7025 подій зменшилася до 3421 за допомогою сортування від нормальних цільових процесів та процесів джерела, а потім від 3421 подій до 610 внаслідок відсіювання за робочими станціями із нормальною діяльністю. Таким чином, був отриманий третій список типу Event. На основі цього списку були повторені пункти алгоритму 3-6 та 9-10 в одній ітерації.

Після застосування до цього списку нового списку виключень за командними рядками було отримано 267 подій, які можна назвати підозрілими та не було виявлено такої діяльності, що була б підтверджена організацією як нормальна та легітимна. Дані події були класифіковані як потенційна APT атака, на основі списку з цими подіями було викликано функцію getTec для отримання

списку типу TechniqueChart із даними для побудови кінцевої хронології подій APT – останнього етапу методу аналізу атак.

На Рис.4.4 показаний кінцевий графік хронології APT із врахуванням того, що існують однакові техніки для декількох тактик одночасно, тому список кінцевих подій був заздалегідь відсортований, щоб дотримуватися поступовості у APT. Кожна точка графіку це унікальна техніка у розрізі однієї дати, тобто насправді у кінцевому сеті даних в один день могло бути застосовано один і більше разів однієї техніки як на одній робочій станції, так і на декількох, що краще відображається у статистиці застосованих технік.

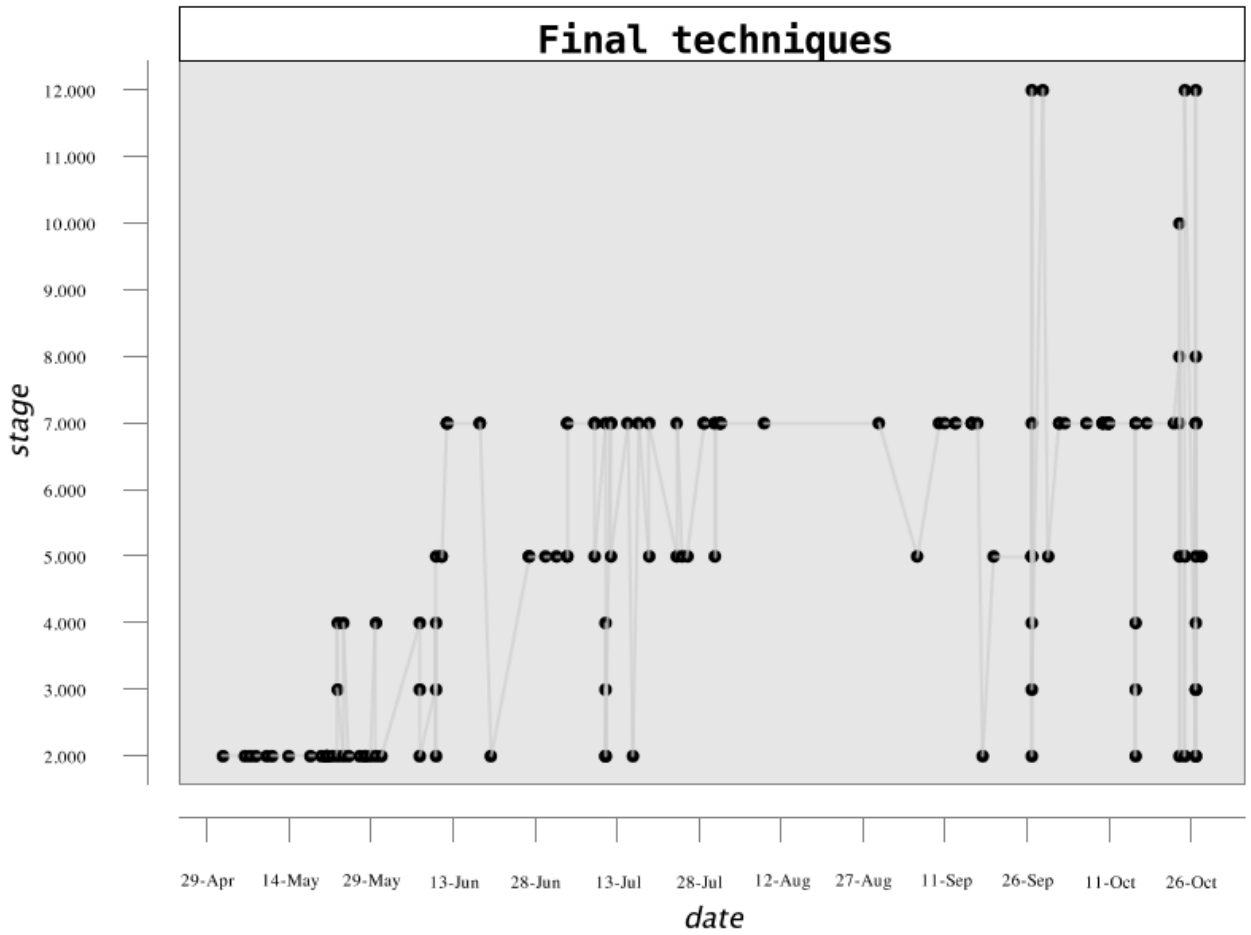

Рисунок 4.4 – Хронологія APT атаки

Останньою дією було вивантаження списку кінцевих даних за допомогою функції getFinalTec на основі останнього аналізованого файлу та попереднього списку типу TechniqueChart. Даний список подій потрібен для відображення послідовності технік та опису їх призначення у розрізі APT.

4.3 Аналіз виявленої APT атаки

В результаті проведення описаного аналізу було отримано список кінцевих подій, що були класифіковані як ключові події проведеної APT.

Перший етап атаки за тактиками «Початковий доступ» за допомогою виявлених технік на основі подій Sysmon та Powershell виявлений не був, але за даними із тестового середовища можна вважати, що зловмисники потрапили всередину мережі шляхом використання техніки Spearphishing Attachment, що є одним із найпоширеніших способів дібратися до жертви.

На стадії «Виконання» зловмисниками була використана техніка T1117 regsvr32 для завантаження та виконання скрипту, який знаходиться на віддаленому сервері – командному центрі. Після виконання скрипту запускався невідомий процес під прихованою назвою calc.exe. Виконання скрипту відбувалося на одному хості soclab-ws01 впродовж декількох днів, а також продублювалося на іншому хості soclab-ws02. Процес regsvr32.exe продовжував з'єднуватися із командним центром. Після цього процесом hh.exe відбувалася компіляція віддаленого файлу із того ж командного центру у рамках техніки T1223 Compiled HTML file. Невідомим процесом calc.exe був ініційований доступ до процесу cmstp.exe та запущений файл розширення .inf у рамках виконання техніки T1191 CSMTP – використано для обходу UAC та обходу блокування запуску усіх процесів, що не входять до білого списку.

Наступна техніка T1170 Mshta була застосована для завантаження скрипту із віддаленого серверу через обхід налаштувань безпеки браузера. Техніка T1085 rundll32 на soclab-ws02 була виконана для запуску завантаженого скрипту javascript. Після чого була виконана T1121 regsvcs-regasm на soclab-ws01 для створення спеціального зловмисного shellcode для подальшого його використання.

T1220 XSL Script Processing та T1047 WMI на soclab-ws01 викликала команду wmic process list за форматом із віддаленого файлу для перевірки наявних запущених процесів. Віддалений файл формату знаходився на

командному центрі 151.101.0.133, до якого відбувалося підключення за портом 443. Знову відбувся виклик невідомого процесу calc.exe.

T1218 Signed Binary Proxy Execution на soclab-ws01 процес powershell.exe виконав команду через параметри -noninteractive -windowstyle hidden -executionpolicy для визначення поточної директорії. Внаслідок цього виклик невідомого процесу calc.exe. T1216 Signed Script Proxy Execution на soclab-ws01 процесом cscript.exe запустили команду для запуску підписаного скрипту PubPrn.vbs із опціями для його завантаження та виконання. Після чого був завантажений у фоновому режимі файл формату .csproj.

Для встановлення 3 тактики «Постійність» (стадії APT) та 4 «Підвищення привілеїв» було здійснено ряд технік для закріплення у системі та встановлення потрібних утиліт, створення нових сервісів та змінення параметрів для запуску певних процесів.

T1127 Trusted Developer Utilities на soclab-ws01 процес msbuild.exe скомпілював завантажений файл формату .csproj. для скомпільованого файлу був створений новий ключ у регістрі, що відобразилося у техніці T1015 Accessibility Features подія реєстру – створений новий ключ.

T1103 AppInit_DLLs на soclab-ws01 відбулася подія реєстру – задано ключ hklm\software\microsoft\windows nt\currentversion\windows\loadappinit_dlls значення dword (0x00000001) для подальшого прихованого завантаження бібліотек при старті системи. T1042 на soclab-ws01 відбулася зміна асоціацій для calc.exe через командний рядок за допомогою параметрів /c assoc. Потім шляхом техніки T1015 процес msmpeng.exe ініціював встановлення параметру debugger у ключі реєстру за допомогою утиліти reg.exe.

Важливим моментом є використання T1035 на soclab-ws01 для створення нового сервісу через sc.exe create sesshijack для становлення rdp із хостом 116, 142 та для soclab-ws02. Після цього техніка T1086 Suspicious powershell була виконана на host116 та host142 для запуску підозрілого скрипту через powershell.exe -executionpolicy bypass -file scr.ps1 унаслідок батьківського

командного рядку gpscript.exe /startup. Створення дочірніх скриптів відбулося шляхом запуску скрипту shellcode.

На 5 стадії «Обхід захисту» були застосовані техніки для обходу UAC та запуску різних скриптів. Для хостів soclab-ws01, soclab-ws02 була виконана техніка T1050 для запуску та створення нових сервісів від скомпільованого файлу. Остання команда також була класифікована як результат виконання технік T1031 Modify Existing Service та T1035 Service Execution. T1088 Bypass UAC на soclab-ws01 процесом reg.exe запущена команда для підміни процесу, що буде запускатися від відкриття блокноту для відкриття командного рядку.

T1179 та T1218 на soclab-ws01 свідчать про виконання mavinject 12928 /injectrunning відносно завантаженої спочатку бібліотеки. T1201 Password Policy Discovery на soclab-ws01 запущено ряд команд за типом net accounts /%група%.

T1086_Suspicious powershell execution на host142 відбувся запуск скрипту -noninteractive -executionpolicy bypass -noprofile -file %logonserver%\netlogon\scr.ps1. запуски скриптів породили збір інформації про мережу та пристроїв, ряд запущених команд вилився у наступні два етапи проведення APT, що описані нижче.

У рамках стадій 6 «Доступ до облікових даних» та 7 «Виявлення» було виконано найбільшу кількість різноманітних технік для вилучення даних про користувачів, системну інформацію, а також мережеву.

Зокрема було використано T1082 System Information Discovery та T1012 Query Registry на soclab-ws01 через команду reg query hklm\system\currentcontrolset\services\disk\enum для визначення усіх пристроїв на хості.

Також було застосовано техніки: T1069 Permission Groups Discovery та T1087 Account Discovery для визначення дозволів для різних груп, T1083 File and Directory Discovery через виконання скриптлету у powershell.exe для збору інформації про файли та директорії на робочій станції із записом цієї інформації в окремий файл, у рамках T1087 запускалися команди для вилучення інформації про користувачів - query user, T1016 System Network Configuration Discovery із

виконанням ряду команд (arp -a, netsh, ipconfig) для розвідування мережевої інформації, T1018 Remote System Discovery та T1135 Network Share Discovery для вивідування віддалених серверів та мережевих шар.

Для стадії 8 «Збір даних» та 10 «Командування і управління» була здійснена T1105 Remote File Copy на soclab-ws02.mytestdomain20190704.local для вилучення із попередніх хостів отриманих даних про систему у вигляді набору файлів (certutil -urlcache -split).

Наступною стадією було «Просочення даних» – 11 етап – за допомогою виконання таких дій: T1105 Remote File Copy, T1002 Data Compressed та T1022 Data Encrypted – пересилання набору файлів у зжатому та зашифрованому вигляді

Останнім етапом був 12 «Вплив», у рамках якого було виконано T1489 Service Stop для зупинки створених сервісів та T1485 для деструкції створених файлів, які залучалися до виконання усіх зловмисних дій.

На основі описаних технік можна зробити висновок, що ціллю цієї потенційної APT атаки було вилучення інформації про користувачів, мережу та пристрої, а також вилучення чутливих файлів із мережі жертви через побудовані канали зв’язку.

Висновки до розділу 4

Розроблений метод аналізу APT атак був відпрацьований поетапно у відповідності із описаними у попередніх розділах реалізаціями кожного із етапів:

1. Фільтрація даних була проведена за допомогою налаштування правил в SIEM ArcSight за допомогою правил для виявлення застосування технік MITRE. Спрацювання правил формували інший тип даних – кейси – які можуть зберігати кореляційні подій (подія, що формується на основі базової події за заданими умовами у правилі SIEM) безстроково.
2. Застосування алгоритму нечіткого пошуку до отриманих даних для побудови взаємозв’язків між процесами був реалізований за допомогою

розробленого застосунку. До отриманих даних було застосовано алгоритми пошуку нормальних та потенційно небезпечних подій, відсіювання таких подій, побудова графів взаємозв’язків між датами, процесами та командними рядками для встановлення порядку дій та їх безпечності чи небезпечності у розрізі організації.

3. Побудова послідовності застосування технік MITRE та співставлення із життєвим циклом APT атаки було реалізовано шляхом створення графіків послідовності подій у програмному застосунку.
4. Визначення хронології подій APT атаки відбувалося шляхом ітеративного відсіювання нормальних подій та визначення пов’язаних технік на різних робочих станціях. Хронологія була побудована за допомогою створення графіку і програмному застосунку.

Із початкового об’єму даних у 10987 подій було отримано 267 подій APT атаки, яка була розгорнута у межах організації. На виході було отримано хронологію подій атаки у вигляді графіку та опис використаних технік у розрізі реалізованої APT атаки, метою якої був збір та вилучення інформації про користувачів, мережу та пристрої, а також вилучення чутливих файлів із мережі жертви через побудовані канали зв’язку.

5 СТАРТАП

У п’ятому розділі проведено розробку стартап-проекту для розробленого методу аналізу APT та його програмної реалізації. Було подано опис ідеї проекту, проведено технічний аудит ідеї, проаналізовано ринкові можливості запуску стартапу, розроблено ринкову стратегію проекту та його маркетингову програму. Результатами даного розділу є опис можливості реалізації цього стартапу та його реальної імплементації.

5.1 Опис ідеї проекту

Ідеєю проекту є створення прикладного методу аналізу APT атак за допомогою застосування технік MITRE та алгоритму Вагнера-Фішера для встановлення хронології подій атаки у подіях операційної системи Windows, а саме програмного застосунку із реалізованим методом за допомогою мови програмування Java із використанням бібліотек, вказаних у попередньому розділі.

Для цього було сформовано опис змісту ідеї проекту, визначено усі можливі потенційні напрями застосування проекту, а також описано основні вигоди для користувача кінцевого продукту у межах визначених напрямів застосування.

Було зроблено висновок, що реалізація даного методу у формі програмної реалізації є унікальною у розрізі підходу до аналізу APT і виявлених методів. Схожих реалізацій та взагалі такого виду методу на ринку виявлено не було, що підтверджує оригінальність та унікальність проекту.

У Таблиці 5.1, яка знаходиться нижче, подано опис ідеї стартап-проекту, а саме: зміст ідеї та можливі базові потенційні ринки (напрями застосування проекту), в межах яких потрібно шукати групи потенційних клієнтів, а також визначено вигоди для користувача продуктом.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Прикладний аналіз APT атак за допомогою технік MITRE та алгоритму нечіткого пошуку, застосованого до кореляційних подій	Компанії, що займаються розслідуванням вторгнень в інформаційні системи та мережі	Проведення послуги із розслідування вторгнень у розрізі розслідування APT атак та чітка побудова хронології подій
	Компанії, що постраждали внаслідок вторгнень	Виявлення слідів вторгнення та компонентів інфраструктури, які було ушкоджено, для отримання рекомендацій із усунення наслідків атаки/вторгнення
	Компанії, що займаються наданням послуг із “Compromise assessment”	Виявлення компонентів інфраструктури, які постраждали внаслідок атаки, для опису наслідків та подальшого детального аналізу

Наступним етапом було проведення аналізу техніко-економічних переваг ідеї порівняно із одним потенційним конкурентом. За переліком техніко-економічних властивостей було визначено ідеї проекту та порівняно із ідеями конкурента. Порівняльний аналіз показників був проведений таким чином, що для реалізації власної ідеї були визначені слабкі, сильні та нейтральні сторони у вигляді текстового опису значень цих параметрів.

Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів було проведено та представлено у Таблиці 5.2 нижче.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1			
1.	Економічні	Реалізація технік MITRE можлива за допомогою існуючих SIEM систем або звичайних створених самописних парсерів подій, тому вартість даного етапу може коливатися приблизно у межах від 0 до 10000$	Ціна за послуги оцінюється в межах від 2 до 6 тисяч доларів	У разі використання платних SIEM систем – велика ціна питання	Універсальність в реалізації	Підходить для реалізації у різних SIEM систем та власних парсерах

Продовження таблиці 5.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конку-рентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1			
2.	Технічні	Аналіз APT атаки на основі запропонованих подій із ОС	Аналіз подій Windows	Об’ємність початкових даних до аналізу	Немає	Унікальність методу для аналізу
3.	Надійності	Обробка наявних подій, що задовольняють указаним типам подій ОС	Обробка основних подій	Якщо відсутні потрібні події, то зменшується покриття техніками	Немає	Безвідмовна обробка подій при правильному налаштуванні логування та парсингу подій
4	Технологічні	Покладення індикаторів на правила та використання методу для аналізу	Застосування внутрішніх методів аналізу	Створення правил,парсинг подій займає тривалий час	Немає	Швидкість роботи та візуалізація результатів

Кінець таблиці 5.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1			
5	Безпеки	Виявлення хронології подій APT	Встановлення хронології подій APT	Об’ємність початкових даних	Повне охоплення аналізом усіх наявних пристроїв	Виявлення потенційно вражених компонентів системи

5.2 Технологічний аудит ідеї проекту

У даному підрозділі було проведено аудит технології для реалізації ідеї. У Таблиці 5.3 було проведено аналіз технологічної здійсненності ідеї проекту.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Покладення технік MITRE на правила кореляції подій	Правила SIEM систем: Arcsight, Splunk, Elasticsearch	Документація щодо побудови правил у певній SIEM системі	Доступна
2	Парсинг подій із формату .csv	Використання бібліотек мов програмування	Java бібліотека OpenCSV	Доступна

Кінець таблиці 5.3

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
3	Побудова пов’язування подій	Використання реалізацій алгоритму на мовах програмування Python, Java, C++ і т.д	Написання алгоритму на мові Java або пошук реалізації у бібліотеках, використання GraphStream	Доступна
4	Побудова хронології подій	Реалізація побудови графіків на мовах програмування Python, Java, C++ і т.д	Використання бібліотеки GraphStream на мові Java	Доступна
Обрана технологія реалізації ідеї проекту: для реалізації даного проекту усі технології доступні, тому було обрано вищеописані технології для мови Java				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

У підрозділі 5.3 було визначено ринкові можливості, що можна використовувати під час ринкового впровадження проекту та загроз, що можуть зашкодити у ході реалізації проекту. Аналіз попиту було проведено у Таблиці 5.4.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	Відсутні
2	Загальний обсяг продаж, грн/ум.од	30 млн. ум. од.
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає

Кінець таблиці 5.4

№	Показники стану ринку	Характеристика
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності по ринку, %	Не менше 120%

За результатами аналізу таблиці зроблено висновок, що ринок є відкритим та стрімко зростаючим, конкуренція відсутня і тому даний ринок можна вважати привабливим для інвестування.

Потенційні групи клієнтів було визначено у Таблиці 5.5 нижче.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Відсутність прикладного методу аналізу APT при швидкому зростанні застосування даних атак у всьому світі	1. Компанії, що займаються розслідуванням кібер-злочинів 2. Компанії, які зазнали APT атаки	Перша група цільових клієнтів має на увазі поширення послуг із розслідування вторгнень, які внаслідок надання послуг отримують грошову вигоду. Друга група прагне вияснити слабкі та вражені місця своєї інфраструктури задля їх усунення, метою цієї групи є посилення захищеності інфраструктури.	Визначення APT атаки та хронології її подій

У Таблиці 5.6 та 5.7 проведено аналіз ринкового середовища у вигляді факторів загроз та факторів можливостей.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока вартість послуги	Вартість ліцензії SIEM систем висока і може зростати	Альтернативний аналіз за допомогою аналізу подій шляхом власних парсерів
2	Поява конкурентних методів на ринку	Створення нових методів і конкуренція зі створеним методом	Покращення методів побудови хронології подій, повна автоматизація процесів
3	Можливість втрати потенційних спрацювань на правилах	Недоліки у методі відбору даних	Покращення фільтрації та відбору подій
4	Встановлення інфляції в країні	Зміни у співвідношенні курсів валют до національної валюти	Пошук альтернативних методів реалізації методу без потрібності великих вкладів

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання кількості реалізованих APT атак	Ріст атак зумовлює попит на їх розслідування	Збільшення кількості споживачів
2	Відсутність наявних конкурентів	Монополізація ринку	Збільшення клієнтів та розширення компанії

Кінець таблиці 5.7

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
3	Поява нових інтеграційних можливостей у SIEM системах	Можливість прямої інтеграції методу у SIEM без залучення сторонніх продуктів	Розробка покращеного методу для інтеграції
4	Розширення супутніх послуг	Надання послуг із аналізу шкідливих файлів, бекдорів, індикаторів компрометації, вразливостей системи	Створення нових послуг

Загальні риси конкуренції на ринку були визначені у Таблиці 5.8, яка розташована нижче.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Вказати тип конкуренції - олігополія	На ринку присутня невелика кількість конкурентів із меншими можливостями	Можливість завоювати ринок внаслідок відсутності схожих методів аналізу
2. За рівнем конкурентної боротьби - національний	Надання послуг на території України та закордоном	Покращення підходів до аналізу, розширення набору послуг

Кінець таблиці 5.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
3. За галузевою ознакою - міжгалузева	Можливість застосування для компаній із різних сфер діяльності	Пошук нових клієнтів серед банківських структур та інших сфер
4. Конкуренція за видами товарів: - товарно-видова	Дана послуга задовольняє потреби споживача, унікальна і відрізняється від конкурентних методів	Надання послуг у сфері інформаційної безпеки
5. За характером конкурентних переваг - цінова	Ключовим фактором для клієнтів є забезпечення захищеності своєї мережі або надання таких послуг, що передбачає зменшення ризиків, витрат на подолання загроз, швидке усунення вразливостей після виявлення вторгнення	Підвищення кості надання послуг та супроводження клієнтів після аналізу
6. За інтенсивністю - марочна	Кінцева привабливість самої послуги	Покращення методу аналізу

Більш детальний аналіз умов конкуренції у галузі було розроблено за моделлю 5 сил М. Портера та його результати подано у Таблиці 5.9, яка подана нижче.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Прямі конкуренти відсутні	Бар'єрами входження в ринок може бути ціна SIEM	Постачальники SIEM	Можуть визнати ефективності даного аналізу для подальшої рекомендації іншим споживачам	Товари-замінники відсутні
Висновки:	Прямі конкуренти відсутні, але є такі, що надають схожі послуги, але на недостатньому рівні, тому вважаються слабкими	Можливість входу в ринок є за умови високого рівня надання послуг та їх ефективності	Напряму не диктує умови роботи на ринку, від нього залежить тільки ціна ліцензії	Клієнти диктують умови роботи на ринку, тому що обрають якість і ціну серед наявних послуг, яка їм більше підходить	Обмеження відсутні

З огляду на конкурентну ситуацію проект має принципові можливості роботи та розвитку на ринку. Його сильними сторонами є унікальність методу аналізу та відсутність потенційних конкурентів, універсальність з огляду на інтеграцію до різних систем аналізу даних. Стрімкий ріст розповсюдження APT атак також грає свою роль у якості сприяння розширення бази клієнтів та розвитку проекту.

У Таблиці 5.10 наведено фактори конкурентоспроможності.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Відсутність прямих конкурентів на ринку надання послуг	Поєднання використання технік MITRE та алгоритму нечіткого пошуку для фільтрації даних та пов’язування подій між собою із подальшою побудовою хронології подій APT
2	Гнучкість методу	Можливість застосування до різних SIEM та аналізаторів подій
3	Застосування для компаній із різних галузей	Аналіз подій не залежить від типу організації і вимагає лише наявність певних типів подій
4	Ціна послуги	Ціна послуги може залежати від платоспроможності клієнта на рахунок використання платних SIEM чи самописних аналізаторів

За визначеними вище факторами конкурентоспроможності було проведено аналіз сильних та слабких сторін стартап-проекту у Таблиці 5.11, яка розташована нижче.

Так як прямих конкурентів немає, то проводилося порівняння із схожими послугами в обраній галузі розробки проекту.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін запропонованого методу

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим методом						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Відсутність прямих конкурентів на ринку надання послуг	15							+
2	Гнучкість методу	18					+		
3	Застосування для компаній із різних галузей	20							+
4	Ціна послуги	15					+		

На основі виділених ринкових загроз та можливостей, слабких та сильних сторін було складено SWOT-аналіз у Таблиці 5.12.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: унікальність методу аналізу, відсутність прямих конкурентів на ринку, гнучкість методу, ціна послуги, застосування для компаній із різних галузей	Слабкі сторони: можлива залежність ціни від компаній-постачальників, залежність від маркетингової схеми
Можливості: розширення кількості споживачів, вихід на міжнародний ринок, розвиток суміжних послуг, створення окремого незалежного продукту для аналізу АРТ на основі створеного методу	Загрози: поява прямих конкурентів, збільшення інфляції, ріст цін від компаній-постачальників

На основі SWOT-аналізу було розроблено альтернативу ринкової поведінки та оптимальний час ринкової реалізації у Таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Повноцінна розробка для однієї SIEM або свого парсеру	Дуже висока	3 місяця
2	Розробка для декількох SIEM	Висока	6 місяців
3	Створення свого продукту для аналізу, незалежного від інших	Висока	1 рік

З огляду на перелічені альтернативи було обрано першу, тому що отримання ресурсів є більш простим і строки реалізації найкоротші із можливих.

5.4 Розроблення ринкової стратегії проекту

Для розроблення ринкової стратегії проекту було початково виконано перший крок – це вибір та опис цільових груп потенційних споживачів продукту, який лежить в основі проекту.

У Таблиці 5.14 описано вибір цільових груп потенційних споживачів.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи	Готовність споживачів сприйняти продукт	Орієнтовний попит	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Компанії великого та середнього бізнесу	Готові	Дуже високий	Немає прямих конкурентів, але є схожі рішення за напрямом аналізу	Простий

Кінець таблиці 5.14

№ п/п	Опис профілю цільової групи	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
2	Компанії до 20 осіб	Готові, але зацікавленість низька	Низький	Немає прямих конкурентів	Простий
3	Поодинокі користувачі та ФОП	Зацікавленість низька	Низький	Немає прямих конкурентів	Простий
Які цільові групи обрано: Компанії великого та середнього бізнесу, тому що мають велику інфраструктуру, якою можуть бути зацікавлені зловмисники					

За результатами аналізу потенційних груп споживачів було обрано одну цільову групу, для якої буде пропонуватися реалізація методу аналізу, тому стратегія охоплення ринку – це стратегія концентрованого маркетингу. Базова стратегія розвитку в обраному сегменті ринку запропонована у Таблиці 5.15.

Таблиця 5.15 – Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
1	Повноцінна розробка для однієї SIEM або свого парсеру	Стратегія концентрованого маркетингу	Задоволення потреб компаній великого, середнього бізнесу краще за конкурентні методи, орієнтація на організації із критичними інфраструктурними одиницями	Стратегія спеціалізації

Стратегія конкурентної поведінки визначена у Таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так	Існуючі споживачі	Ні	Стратегія лідера

На основі вимог користувачів із обраної групи до постачальника кінцевого продукту проекту було розроблено стратегію позиціонування. У Таблиці 5.17 подана дана стратегія.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Пошук APT атаки у подіях, аналіз атаки та побудова хронології подій	Стратегія спеціалізації	Пошук APT атаки у подіях за допомогою технік, аналіз атаки та побудова хронології подій із використанням алгоритму нечіткого пошуку	Аналіз APT атак Побудова хронології APT Визначення вражених елементів інфраструктури

5.5 Розроблення маркетингової програми стартап-проекту

Першим етапом даного підрозділу була розробка та формування концепції кінцевого товару (продукту), яка формувалася на основі проаналізованої конкурентоспроможності цього товару.

У Таблиці 5.18 описано маркетингова концепція товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Знаходження APT атаки у подіях	Скорочення аналізованих подій у 3000 разів	Використання подій Sysmon та Powershell для аналізу
2	Аналіз APT	Аналіз шляхом використання популярної бази знань MITRE	Використання технік MITRE
3	Побудова хронології APT	Пошук взаємопов’язаних процесів як на хості, так і у мережі	Застосування алгоритму нечіткого пошуку
4	Знайдення вражених елементів інфраструктури	Покриття аналізом усієї інфраструктури	Застосування алгоритму нечіткого пошуку та пов’язування подій

Далі була розроблена трирівнева маркетингова модель товару із уточненням усіх важливих факторів та побудована у Таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Прикладний аналіз APT атак за допомогою технік MITRE та алгоритму нечіткого пошуку, застосованого до кореляційних подій</td></tr>
<tr><td rowspan="8">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Аналіз подій</td><td>М</td><td>Тх, Тл, Е</td></tr>
<tr><td>2.Повязування подій</td><td>М</td><td>Тх</td></tr>
<tr><td>3.Побудова хронології подій</td><td>М</td><td>Тх, Е</td></tr>
<tr><td>4.Фільтрація подій</td><td>М</td><td>Тх, Е</td></tr>
<tr><td>5.Відсіювання фальш-позитиву</td><td>М</td><td>Тх, Тл, Е</td></tr>
<tr><td colspan="3">Якість: ISO/IEC 27001, ISO/IEC 27002</td></tr>
<tr><td colspan="3">Марка: APT-Hunter</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: демонстрація можливості виявлення застосування технік</td></tr>
<tr><td colspan="3">Після продажу: прикладний аналіз подій та пошук APT</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, шифрування, унікальний підхід до побудови хронології подій</td></tr>
</table>

У Таблиці 5.20 було визначено межі встановлення цін на потенційний товар.

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Відсутній	10000-15000$	Високий	Від 8000$ до 15000$

Оптимальна система збуту, в межах якої приймається рішення, подана у Таблиці 5.21.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту	Глибина каналу збуту	Оптимальна система збуту
1	Невелика кількість посередників	Якісна маркетингова політика та її постійне підвищення, пошук нових каналів збуту	Канали нульового рівня та першого рівня	Збут через посередників та безпосередньо до групи споживачів напряму

Концепція маркетингових комунікацій описана у Таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій цільових клієнтів	Ключові позиції для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Демонстрація раніше виявлених атак за допомогою методу	Соціальні мережі, конференції, хакатони	Аналіз APT атак та побудова хронології подій	Показати масштаб потенційних втрат від реалізованої атаки та низьку імовірність знаходження усіх вражених елементів інфраструктури	Демонстрація виявлення APT та відображення її хронології у повному масштабі

Висновки до розділу 5

На даний момент динаміка ринку така, що постійно зростає попит на послугу аналізу APT атак, що напряму залежить від стрімкого зростання популяризації даної загрози в інформаційному просторі. Тому комерціалізація даного проекту є цілком можливою. Перспективи впровадження, перш за все, визначені для однієї цільової групи – компанії великого та середнього бізнесу, які потенційно обираються цілями для здійснення APT.

Наразі прямих конкурентів немає, але є схожі послуги із розслідування APT атак на ринку, хоча вони проводяться не на достатньо високому рівні та можуть бути не цілком ефективними. Бар’єрами входження в ринок може бути ціна SIEM, якщо використовувати її як основу для реалізації правил на основі технік. Факторами конкурентоспроможності проекту є: відсутність прямих конкурентів на ринку надання послуг, гнучкість методу, застосування для компаній із різних галузей, ціна та якість послуги.

Альтернативою впровадження доцільно обрати повноцінну розробку для однієї SIEM або свого парсеру.

Подальша імплементація проекту є цілком доцільною та реальною.

ВИСНОВКИ

Метою даної роботи була розробка методу аналізу APT атаки за допомогою технік MITRE та із застосуванням алгоритму нечіткого пошуку. Етапами даного методу були визначені такі дії:

1. Фільтрація даних за допомогою технік MITRE.
2. Побудова взаємозв’язків між процесами за допомогою алгоритму нечіткого пошуку.
3. Побудова послідовності технік за життєвим циклом APT атаки.
4. Визначення кінцевої хронології подій APT атаки.

Внаслідок проведення ряду робіт із аналізу існуючих методів аналізу даного типу атак було вирішено покласти в основу нового методу аналізу матрицю MITRE, на основі якої провести тестування технік та вилучити індикатори їх застосування серед подій Sysmon та Powershell. Завдяки отриманим подіям було створено список індикаторів для технік у розрізі кожної із проаналізованих тактик та розроблено набір правил для спрацювання у SIEM ArcSight. Дані роботи проводилися у розрізі етапу фільтрації даних. Даними для аналізу слугували дані організації, для якої було проведено створення та налаштування правил на основі індикаторів технік MITRE для SIEM ArcSight, було проведено аналіз потенційної APT.

Потреба у використанні алгоритму нечіткого пошуку виникла на етапі знайдення подій, що пов’язані між собою за певними параметрами, та для побудови зв’язків між параметрами для подій однієї чи декількох робочих станцій. Проблема пошуку за допомогою хешів процесів є неефективною у даному випадку, тому що існують похибки у можливих відмінностях у місцезнаходженнях процесів, у версіях процесів, у навмисно змінених назвах процесів, у параметрах командних рядків. Тому для вирішення цієї проблеми було використано алгоритм нечіткого пошуку Вагнера-Фішера, що дозволяє обчислити

рівень схожості між двома рядковими даними та на основі обчисленого значення робити висновок про те, чи можна вважати ці рядки однією сутністю.

Для реалізації 2-4 етапів методу був розроблений спеціальний застосунок мовою програмування Java із використанням бібліотек XChart, Graphstream, Apache CSV для забезпечення зчитування даних та їх подальшого графічного відображення згідно із потребами методу.

До отриманих даних організації було застосовано алгоритми пошуку нормальних та потенційно небезпечних подій, відсіювання таких подій, побудова графів взаємозв'язків між датами, процесами та командними рядками для встановлення порядку дій та їх безпечності чи небезпечності у розрізі організації.

Побудова послідовності застосування технік MITRE за життєвим циклом APT було реалізовано шляхом відтворення графіків із послідовністю подій на основі ідентифікаторів технік.

Визначення хронології подій APT атаки відбувалося шляхом ітеративного відсіювання нормальних подій та визначення пов'язаних технік на різних робочих станціях. Хронологія APT була створена шляхом побудови графіку у програмному застосунку.

Із початкового об'єму даних у 10987 подій було отримано 267 подій APT атаки, яка була розгорнута у межах організації. На виході було отримано хронологію подій APT у вигляді графіку та опис використаних технік у розрізі реалізованої атаки, метою якої був збір та вилучення інформації про користувачів, мережу та пристрої, а також вилучення чутливих файлів із мережі жертви через побудовані канали зв'язку.

ПЕРЕЛІК ПОСИЛАНЬ

1 Anatomy of an APT Attack: Step by Step Approach [Електронний ресурс]. – Режим доступу: https://resources.infosecinstitute.com/anatomy-of-an-apt-attack-step-by-step-approach/#gref (дата звернення: 02.09.2019).

2 Defending Against Advanced Persistent Threats [Електронний ресурс]. – Режим доступу: https://www.necam.com/docs/?id=759a3111-a395-4c40-ae0a-43e299159c81(дата звернення: 02.09.2019).

3 NIST. Managing Information Security Risk: Organization, Mission, and Information System View Approach [Електронний ресурс]. – Режим доступу: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=908030 (дата звернення: 05.09.2019).

4 Advanced Persistent Threat (APT) [Електронний ресурс]. – Режим доступу: https://www.incapsula.com/web-application-security/apt-advanced-persistent-threat.html (дата звернення: 10.09.2019).

5 Advanced Persistent Threats: A Symantec Perspective. Preparing the Right Defense for the New Threat Landscape [Електронний ресурс]. – Режим доступу: https://www.symantec.com/content/en/us/enterprise/white_papers/b-advanced_persistent_threats_WP_21215957.en-us.pdf (дата звернення: 11.09.2019).

6 The Diamond Model For Intrusion Analysis: A Primer, Andy Pendergast, 2014 [Електронний ресурс]. – Режим доступу: http://www.activeresponse.org/wp-content/uploads/2013/07/diamond.pdf (дата звернення: 04.09.2019).

7 The Cyber Kill Chain Explained [Електронний ресурс]. – Режим доступу: https://www.forbes.com/sites/forbestechcouncil/2018/10/05/the-cyber-kill-chain-explained/#1b7c82f26bdf (дата звернення: 04.09.2019).

8 MITRE [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/ (дата звернення: 02.09.2019).

9 Atomic Red Team [Електронний ресурс]. – Режим доступу: https://atomicredteam.io/ (дата звернення: 03.09.2019).

10 Фролов А. С. Розробка алгоритму нечіткого пошуку на основі хешування // Молодий вчений. - 2016. - №13. - С. 357-360. [Електронний ресурс]. – Режим доступу: https://moluch.ru/archive/117/32158/ (дата звернення: 08.10.2019).

11 Желудков, А. В., Макаров, Д. В., Фадеев, П. В. Особенности алгоритмов нечёткого поиска. Москва, Инженерный вестник МГТУ им. Н.Э. Баумана, 2014. – С. 502-503

12 Soundex [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Soundex (дата звернення: 08.10.2019).

13 Поиск на неточное соответствие: коды Хемминга [Електронний ресурс]. – Режим доступу: http://www.jurnal.org/articles/2009/inf32.html (дата звернення: 09.10.2019).

14 Двоичный алгоритм поиска подстроки [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Двоичный_алгоритм_поиска_подстроки (дата звернення: 09.10.2019).

15 Задача о редакционном расстоянии, алгоритм Вагнера-Фишера [Електронний ресурс]. – Режим доступу: http://neerc.ifmo.ru/wiki/index.php?title=Задача_о_редакционном _расстоянии,_алгоритм_Вагнера-Фишера (дата звернення: 10.10.2019).

16 Расстояние Дамерау — Левенштейна [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Расстояние_Дамерау_—_Левенштейна (дата звернення: 10.10.2019).

17 Двоичные коды с исправлением выпадений, вставок и замещений символов [Електронний ресурс]. – Режим доступу: http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=dan&paperid=31411&option_lang=rus (дата звернення: 10.10.2019).

18 Анализ строк [Электронный ресурс]. – Режим доступа: http://itman.narod.ru/articles/infoscope/string_search.1-3.html (дата звернення: 15.10.2019).

19 Windows Matrix [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/matrices/enterprise/windows/ (дата звернення: 02.09.2019).

20 Atomics [Електронний ресурс]. – Режим доступу: https://github.com/redcanaryco/atomic-red-team/tree/master/atomics (дата звернення: 02.10.2019).

21 Sysmon v10.41 [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon (дата звернення: 20.09.2019).

22 PowerShell Commands Cheat Sheet [Електронний ресурс]. – Режим доступу: https://www.comparitech.com/net-admin/powershell-cheat-sheet/ (дата звернення: 23.09.2019).

ДОДАТОК А

Програмний код реалізованого методу:

Techniques.java

```
package fuzzystringsearch;

/**
 *
 * @author ira
 */
public class Techniques {
        private int id;
        private String idTec;
    private String Name;

    public int getId() {
        return id;
    }
        public void setId(int id) {
        this.id = id;
    }
        public String getIdTec() {
        return idTec;
    }
        public void setIdTec(String idTec) {
        this.idTec = idTec;
    }
    public String getName() {
        return Name;
    }
        public void setName(String Name) {
        this.Name = Name;
    }

    @Override
    public String toString(){
        return "ID of tactic="+id+",ID of technique="+idTec+",Name="+Name+"\n";
    }
}
```

Tactics.java

```
package fuzzystringsearch;

/**
 *
 * @author ira
 */
public class Tactics {
        private int id;
    private String Tactic;
        private String Name;

    public int getId() {
        return id;
    }
        public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return Name;
    }
        public void setName(String Name) {
        this.Name = Name;
    }

    public String getTactic() {
        return Tactic;
    }
        public void setTactic(String Tactic) {
        this.Tactic = Tactic;
    }

        @Override
    public String toString(){
        return "ID="+id+",Name="+Name+",Tactic="+Tactic+"\n";
    }

}
```

Levenstein.java

```
package fuzzystringsearch;

import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author ira
 */
public class Levenstein {
    public static int distance(String a, String b) {
        a = a.toLowerCase();
        b = b.toLowerCase();
        // i == 0
        int [] costs = new int [b.length() + 1];
        for (int j = 0; j < costs.length; j++)
            costs[j] = j;
        for (int i = 1; i <= a.length(); i++) {
            // j == 0; nw = lev(i - 1, j)
            costs[0] = i;
            int nw = i - 1;
            for (int j = 1; j <= b.length(); j++) {
                int cj = Math.min(1 + Math.min(costs[j], costs[j - 1]),
                a.charAt(i - 1) == b.charAt(j - 1) ? nw : nw + 1);
                nw = costs[j];
                costs[j] = cj;
            }
        }
        return costs[b.length()];
    }

}
```

IdTechniques.java

```java
package fuzzystringsearch;

/**
 *
 * @author ira
 */
public class IdTechniques {
        private String id;
    private String Name;

    public String getId() {
        return id;
    }
        public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return Name;
    }
        public void setName(String Name) {
        this.Name = Name;
    }

    @Override
    public String toString(){
        return "ID="+id+",Name="+Name+"\n";
    }
}
```

Event.java

```
package fuzzystringsearch;
/**
 *
 * @author ira
 */
public class Event {
    private String id;
    private String Name;
    private String Date;
    private String Time;
    private String DeviceHostName;
    private String SourceProcessName;
    private String DestinationAddress;
    private String DestinationPort;
    private String DestinationUserName;
    private String DestinationProcessName;
    private String FilePath;
    private String FileHash;
    private String DC1;
    private String DC2;
    private String DC3;
    private String DC5;
    private String DC6;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return Name;
    }
    public void setName(String Name) {
        this.Name = Name;
    }
    public String getTime() {
        return Time;
    }
    public void setTime(String Time) {
        this.Time = Time;
    }
    public String getDate() {
        return Date;
    }
    public void setDate(String Date) {
        this.Date = Date;
    }
    public String getDeviceHostName() {
        return DeviceHostName;
    }
```

```
public void setDeviceHostName(String DeviceHostName) {
    this.DeviceHostName = DeviceHostName;
}
public String getSourceProcessName() {
    return SourceProcessName;
}
public void setSourceProcessName(String SourceProcessName) {
    this.SourceProcessName = SourceProcessName;
}
public String getDestinationAddress() {
    return DestinationAddress;
}
public void setDestinationAddress(String DestinationAddress) {
    this.DestinationAddress = DestinationAddress;
}
public String getDestinationPort() {
    return DestinationPort;
}
public void setDestinationPort(String DestinationPort) {
    this.DestinationPort = DestinationPort;
}
public String getDestinationUserName() {
    return DestinationUserName;
}
public void setDestinationUserName(String DestinationUserName) {
    this.DestinationUserName = DestinationUserName;
}
public String getDestinationProcessName() {
    return DestinationProcessName;
}
public void setDestinationProcessName(String DestinationProcessName) {
    this.DestinationProcessName = DestinationProcessName;
}
public String getFilePath() {
    return FilePath;
}
public void setFilePath(String FilePath) {
    this.FilePath = FilePath;
}
public String getFileHash() {
    return FileHash;
}
public void setFileHash(String FileHash) {
    this.FileHash = FileHash;
}
public String getDC1() {
    return DC1;
}
public void setDC1(String DC1) {
    this.DC1 = DC1;
}
public String getDC2() {
    return DC2;
}
```

```
public void setDC2(String DC2) {
    this.DC2 = DC2;
}
public String getDC3() {
    return DC3;
}
public void setDC3(String DC3) {
    this.DC3 = DC3;
}
public String getDC5() {
    return DC5;
}
public void setDC5(String DC5) {
    this.DC5 = DC5;
}
public String getDC6() {
    return DC6;
}
public void setDC6(String DC6) {
    this.DC6 = DC6;
}

@Override
public String toString(){
    return "ID="+id+",Name="+Name+",DeviceHostName="+DeviceHostName+",Date="+Date+"\n";
}
```

TechniqueChart.java

```
package fuzzystringsearch;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 *
 * @author ira
 */
public class TechniqueChart {
        private int id;
        private String date;
        private String Name;
        private String concat;
        private String idDate;
    public int getId() {
        return id;
    }
        public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return Name;
    }
        public void setName(String Name) {
        this.Name = Name;
    }
    public String getDate() {
        return date;
    }
        public void setDate(String date) {
        this.date = date;
    }
        public String getConcat() {
        return concat;
    }
        public void setConcat(String concat) {
        this.concat = concat;
    }
        public String getiddate() {
        return idDate;
    }
        public void setiddate(String idDate) {
        this.idDate = idDate;
    }
        @Override
    public String toString(){
        return "ID="+id+",Name="+Name+",Date="+date+"\n";
    }
}
```

ParserCsv.java

```
import static fuzzystringsearch.Levenstein.distance;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.stream.Collectors;

public class ParserCsv  {

        public List<Event> parsingAll () throws IOException {
            List<Event> emps = new ArrayList();
            CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
            CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/th_events_all_rows.csv"), format);
            for(CSVRecord record : parser){
            Event event = new Event();
            event.setId(record.get("ID"));
            event.setName(record.get("Name"));
                    event.setDate(record.get("Date"));
            event.setTime(record.get("Time"));
                    event.setDeviceHostName(record.get("Event-Device Host Name"));
            event.setSourceProcessName(record.get("Event-Source Process Name"));
            event.setDestinationAddress(record.get("Event-Destination Address"));
            event.setDestinationPort(record.get("Event-Destination Port"));
                    event.setDestinationUserName(record.get("Event-Destination User Name"));
            event.setDestinationProcessName(record.get("Event-Destination Process Name"));
            event.setFilePath(record.get("Event-File Path"));
            event.setFileHash(record.get("Event-File Hash"));
                    event.setDC1(record.get("Event-Device Custom String1"));
            event.setDC2(record.get("Event-Device Custom String2"));
            event.setDC3(record.get("Event-Device Custom String3"));
                    event.setDC5(record.get("Event-Device Custom String5"));
            event.setDC6(record.get("Event-Device Custom String6"));
            emps.add(event);
        }
        parser.close();
            return emps;
        }

 public List<Event> parsingAll2 () throws IOException {
     List<Event> emps = new ArrayList();
     CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
     CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/new_all_rows.csv"), format);
     for(CSVRecord record : parser){
     Event event = new Event();
     event.setId(record.get("ID"));
     event.setName(record.get("Name"));
                 event.setDate(record.get("Date"));
     event.setTime(record.get("Time"));
                 event.setDeviceHostName(record.get("Event-Device Host Name"));
     event.setSourceProcessName(record.get("Event-Source Process Name"));
     event.setDestinationAddress(record.get("Event-Destination Address"));
     event.setDestinationPort(record.get("Event-Destination Port"));
                 event.setDestinationUserName(record.get("Event-Destination User Name"));
     event.setDestinationProcessName(record.get("Event-Destination Process Name"));
     event.setFilePath(record.get("Event-File Path"));
     event.setFileHash(record.get("Event-File Hash"));
                 event.setDC1(record.get("Event-Device Custom String1"));
     event.setDC2(record.get("Event-Device Custom String2"));
     event.setDC3(record.get("Event-Device Custom String3"));
                 event.setDC5(record.get("Event-Device Custom String5"));
     event.setDC6(record.get("Event-Device Custom String6"));
     emps.add(event);
 }
     parser.close();
     return emps;
 }
```

```
public List<Event> parsingAllCmdExc (List<Event> emps,  List<String> cmd) throws IOException {
        List<Event> emps1=emps;
        List<Event> list=new ArrayList();
        for(Event record : emps1){
            for(String val : cmd){
            if(record.getDC1().equals(val) ){
                list.add(record);
            }
            }
    }
    emps1.removeAll(list);
    return emps1;
}

public List<Event> parsingAllHostExc (List<Event> emps,  List<String> host) throws IOException {
    List<Event> res=emps;
    List<Event> list=new ArrayList();
    for(Event record : res){
        for(String val : host){
        if(record.getDeviceHostName().contains(val) ){
                list.add(record);
        }
    }
    }
    res.removeAll(list);
    return res;
}


public List<String> parsingHostExc () throws IOException {
    List<String> host = new ArrayList();
    CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
    CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/hosts_exc.csv"), format);
    for(CSVRecord record : parser){
                String k = record.get(0);
    host.add(k);
}
parser.close();
        return host;
}

public List<String> parsingCmdExc () throws IOException {
    List<String>  cmd= new ArrayList();
    CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
    CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/cmd_exc.csv"), format);
    for(CSVRecord record : parser){
                String k = record.get(0);
    cmd.add(k);
}
parser.close();
        return cmd;
}

public List<IdTechniques> parsingTechniques () throws IOException {
    List<IdTechniques> idtec = new ArrayList();
    CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
    CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/all_techniques.csv"), format);
    for(CSVRecord record : parser){
    IdTechniques event = new IdTechniques();
    event.setId(record.get("ID"));
    event.setName(record.get("Name"));
    idtec.add(event);
}
parser.close();
        return idtec;
}
```

```
public List<Tactics> parsingTactices () throws IOException {
    List<Tactics> tec = new ArrayList();
    CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
    CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/matrix_by_tactics.csv"), format);
    for(CSVRecord record : parser){
    Tactics event = new Tactics();
    event.setTactic(record.get("Tactic"));
                event.setId(Integer.valueOf(record.get("TacticId")));
                event.setName(record.get("Name"));
                tec.add(event);
    }
    parser.close();
    return tec;
}

public List<Techniques> createIdTecList (List<IdTechniques> idtec, List<Tactics> tec) throws IOException {
    List<Techniques> all = new ArrayList();
    for(Tactics ev : tec){
            for(IdTechniques ev2 : idtec){
                if (ev.getName().equals(ev2.getName())){
                Techniques event = new Techniques();
    event.setId(Integer.valueOf(ev.getId()));
    event.setName(ev.getName());
                event.setIdTec(ev2.getId());
    all.add(event);
                }
            }
        }
        return all;
}

public int countIPFreq (List<Event> emps, int n){
    for (Event ev : emps){
        if (ev.getDestinationAddress()!=null){
            n++;
        }
    }
    return n;
}


public List<String> getDestIpNotNull (List<Event> emps, List<String> list){
    for (Event ev : emps){
        if (ev.getDestinationAddress()!=null){
            System.out.println("IP address: " +ev.getDestinationAddress()+
                    "     UserName:    "+ev.getDestinationUserName()+"      "
                    +ev.toString());
            list.add(ev.getDestinationAddress());
        }
    }
    return list;
}

 public List<String> getDestIpNotNullHost (List<Event> emps){
     List<String> host = new ArrayList();
     for (Event ev : emps){
         if (ev.getDestinationAddress()!=null){
             if(!host.contains(ev.getDeviceHostName())){
                 host.add(ev.getDeviceHostName());
             }
         }
     }
     return host;
 }

 public List<Event> getEventsHost (List<Event> emps, String host){
     List<Event> listHost=new ArrayList();
     for (Event record : emps){
         if (record.getDeviceHostName().equals(host)){
             listHost.add(record);
         }
     }
     return listHost;
 }
```

```
public void getCmdForHost(List<Event> emps, List<String> host){
    for (Event ev : emps){
        for (String h : host){
            if (ev.getDeviceHostName().equals(h)){
                System.out.println("Device Host Name :  "+h+"    "
                        +"CMD: "+ev.getDC1()+"\n");
            }
        }
    }
}

public Map<String, Integer> countFrequencies(List<String> list)
{
    Map<String, Integer> hm = new HashMap<>();

    for (String i : list) {
        Integer j = hm.get(i);
        hm.put(i, (j == null) ? 1 : j + 1);
    }
    return hm;
}

public Map<String, Integer> sortbyvalue(Map<String, Integer> map)
{
    return map.entrySet()
        .stream()
        .sorted((Map.Entry.<String, Integer>comparingByValue().reversed()))
        .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue,
                (e1, e2) -> e1, LinkedHashMap::new));

}

public void showMap(Map<String, Integer> map){
    map.entrySet().forEach((entry) -> {
        String k = entry.getKey();
        int v = entry.getValue();
        System.out.println("Value:  "+v+"  cmd:     "+k);
    });
}

public List<String> parsingDate (List<Event> emps) throws IOException {
        List<String> date = new ArrayList();
        for(Event record : emps){
            if(!date.contains(record.getDate())) {
                date.add(record.getDate());
            }
        }
        return date;
}

public List<String> parsingDateForHost (List<Event> emps, String host) throws IOException {
    List<String> date = new ArrayList();
        for(Event record : emps){
            if(!date.contains(record.getDate()) &&
                    record.getDeviceHostName().equals(host)) {
                date.add(record.getDate());
            }
        }
        return date;
}

public List<String> parsingTPN (List<Event> emps) throws IOException {
    List<String> tpn = new ArrayList();
        for(Event record : emps){
            tpn.add(record.getDestinationProcessName());
        }
        return tpn;
}
```

```
public List<String> parsingUniqueTPN (List<Event> emps) throws IOException {
    List<String> tpn = new ArrayList();
        for(Event record : emps){
            if(!tpn.contains(record.getDestinationProcessName())) {
                tpn.add(record.getDestinationProcessName());
            }
        }
        return tpn;
}

public List<String> parsingUniqueTPNForHost (List<Event> emps, String host) throws IOException {
    List<String> tpn = new ArrayList();
        for(Event record : emps){
            if(!tpn.contains(record.getDestinationProcessName())
                    && record.getDeviceHostName().equals(host)) {
                tpn.add(record.getDestinationProcessName());
            }
        }
        return tpn;
}

public List<String> parsingHost (List<Event> emps) throws IOException {
    List<String> host = new ArrayList();
        for(Event record : emps){
            host.add(record.getDeviceHostName());
        }
        return host;
}

public List<String> parsingUniqueHost (List<Event> emps) throws IOException {
    List<String> host = new ArrayList();
        for(Event record : emps){
            if(!host.contains(record.getDeviceHostName())) {
                host.add(record.getDeviceHostName());
            }
        }
        return host;
}

public List<String> parsingCmd (List<Event> emps) throws IOException {
    List<String> cmd = new ArrayList();
        for(Event record : emps){
            cmd.add(record.getDC1());
        }
        return cmd;
}

public List<String> parsingUniqueCmd (List<Event> emps) throws IOException {
    List<String> cmd = new ArrayList();
        for(Event record : emps){
            if(!cmd.contains(record.getDC1())) {
                cmd.add(record.getDC1());
            }
        }
        return cmd;
}

public List<String> parsingUniqueCmdForHost (List<Event> emps, String host) throws IOException {
    List<String> cmd = new ArrayList();
        for(Event record : emps){
            if(!cmd.contains(record.getDC1())
                    && record.getDeviceHostName().equals(host)) {
                cmd.add(record.getDC1());
            }
        }
        return cmd;
}
```

```
public List<String> parsingSPN (List<Event> emps) throws IOException {
    List<String> spn = new ArrayList();
        for(Event record : emps){
            spn.add(record.getSourceProcessName());
        }
        return spn;
}

public List<String> parsingHost (List<Event> emps) throws IOException {
    List<String> host = new ArrayList();
        for(Event record : emps){
            host.add(record.getDeviceHostName());
        }
        return host;
}

public List<String> parsingUniqueHost (List<Event> emps) throws IOException {
    List<String> host = new ArrayList();
        for(Event record : emps){
            if(!host.contains(record.getDeviceHostName())) {
                host.add(record.getDeviceHostName());
            }
        }
        return host;
}

public List<String> parsingCmd (List<Event> emps) throws IOException {
    List<String> cmd = new ArrayList();
        for(Event record : emps){
            cmd.add(record.getDC1());
        }
        return cmd;
}

public List<String> parsingUniqueCmd (List<Event> emps) throws IOException {
    List<String> cmd = new ArrayList();
        for(Event record : emps){
            if(!cmd.contains(record.getDC1())) {
                cmd.add(record.getDC1());
            }
        }
        return cmd;
}

public List<String> parsingUniqueCmdForHost (List<Event> emps, String host) throws IOException {
    List<String> cmd = new ArrayList();
        for(Event record : emps){
            if(!cmd.contains(record.getDC1())
                    && record.getDeviceHostName().equals(host)) {
                cmd.add(record.getDC1());
            }
        }
        return cmd;
}



public List<String> parsingSPN (List<Event> emps) throws IOException {
    List<String> spn = new ArrayList();
        for(Event record : emps){
            spn.add(record.getSourceProcessName());
        }
        return spn;
}
```

```
public List<Event> getFinalTec(List<Event> emps, List<TechniqueChart> ch)throws IOException {
    List<Event> list = new ArrayList();
    for(Event record : emps){
        for(TechniqueChart ev : ch){
            if(record.getName().contains(ev.getName())
                    && record.getDate().equals(ev.getDate())
                    && !list.contains(record)){
                list.add(record);
            }
        }
    }
    return list;
}

public List<TechniqueChart> getTec(List<Event> emps,List<Techniques> all){
    List<TechniqueChart> ch = new ArrayList();
    for(Event ev : emps){
            for(Techniques ev2 : all){
                if (ev.getName().contains(ev2.getIdTec())){
                TechniqueChart event = new TechniqueChart();
                event.setId(ev2.getId());
    event.setDate(ev.getDate());
    event.setName(ev2.getIdTec());
                event.setConcat(ev.getDate()+" / "+ev2.getIdTec());
                event.setiddate(ev2.getIdTec()+" / "+ev2.getId()+" / "+ev.getDate());
                ch.add(event);
                }
            }
    }
    return ch;
}
```

FuzzyStringSearch.java

```
package fuzzystringsearch;

import java.io.IOException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;

/**
 *
 * @author ira
 */
public class FuzzyStringSearch {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException, ParseException {
            ParserCsv csv = new ParserCsv();
            List<Event> n = csv.parsingAll2();
            System.out.println(n.size());
            List<Event> emps = csv.parsingAll();
            System.out.println(emps.size());
            List<String> cmdExc = csv.parsingCmdExc();
            List<Event> emps1 = csv.parsingAllCmdExc(emps, cmdExc);
            System.out.println(emps1.size());
            List<String> hostExc = csv.parsingHostExc();
            List<Event> emps2 = csv.parsingAllHostExc(emps1, hostExc);
            System.out.println(emps2.size());
            List<String> date = csv.parsingDate(emps);
            List<String> tpn = csv.parsingUniqueTPN(emps);
            List<String> host = csv.parsingHost(emps);
            List<String> hostU = csv.parsingUniqueHost(emps);
            List<String> cmd = csv.parsingCmd(emps);
            List<Tactics> tec = csv.parsingTactices();
            List<IdTechniques> idtec = csv.parsingTechniques();
            List<Techniques> all = csv.createIdTecList(idtec, tec);
            TechniqueChart cha = new TechniqueChart();
            List <TechniqueChart> r = cha.getTec(emps2, all);
            Chart chart4 = new Chart();
            List <TechniqueChart> ch = new ArrayList<>();
            chart4.getChart(n, all, "All events");
            for(String h : hostU){
                EventGraph graph = new EventGraph();
                Chart chart1 = new Chart();
                List<Event> eh =   csv.getEventsHost(emps2, h);
                List <TechniqueChart> ch1 = new ArrayList<>();
                List<String> hdate = csv.parsingDateForHost(eh, h);
                List<String> htpn = csv.parsingUniqueTPNForHost(eh, h);
                List<String> hcmd = csv.parsingUniqueCmdForHost(eh, h);
                chart1.getChart(eh, all, ch1, h);
                chart1.getBubbleChart(eh, all, ch1, h);
                graph.graphEventTrio(eh, hdate, htpn, h);
            }
    }
}
```

EventGraph.java

```
package fuzzystringsearch;

import static fuzzystringsearch.Levenstein.distance;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;
import java.util.Map;
import org.graphstream.graph.EdgeRejectedException;
import org.graphstream.graph.Graph;
import org.graphstream.graph.IdAlreadyInUseException;
import org.graphstream.graph.Node;
import org.graphstream.graph.implementations.MultiGraph;
import org.graphstream.ui.swingViewer.ViewPanel;
import org.graphstream.ui.view.Viewer;

/**
 *
 * @author ira
 */
public class EventGraph {
    public void graphEventDateTPN(List<Event> emps, List<String> date, List<String> tpn, String name)
    throws NullPointerException, EdgeRejectedException, IdAlreadyInUseException, FileNotFoundException, IOException {
        Graph graph = new MultiGraph("Graph2");
        graph.addAttribute("ui.quality");
        graph.addAttribute("ui.antialias");
        Viewer viewer = new Viewer(graph, Viewer.ThreadingModel.GRAPH_IN_ANOTHER_THREAD);
        ViewPanel viewPanel = viewer.addDefaultView(false);
        viewPanel.getCamera().setViewPercent(1);
        viewPanel.getCamera().setViewCenter(0, 0, 0);
        for (int i = 0; i < date.size(); i++){
            graph.addNode(date.get(i));
        }
        for (int i = 0; i < tpn.size(); i++){
            graph.addNode(tpn.get(i));
        }
        for (int i = 0; i < date.size(); i++){
             for (int j = 0; j < emps.size(); j++){
                if (distance(date.get(i), emps.get(j).getDate())==0){
                    if (graph.getEdge("connected "+emps.get(j).getDate()+emps.get(j).getDestinationProcessName())==null){
                        graph.addEdge("connected "+emps.get(j).getDate()+emps.get(j).getDestinationProcessName(),
                        date.get(i), emps.get(j).getDestinationProcessName());}
                }
            }
        }
        for (Node node : graph) {
            node.addAttribute("ui.label", node.getId());
        }
        graph.addAttribute("ui.stylesheet", "node {size:3px;fill-color:#777;z-index:0;}");
        graph.addAttribute("ui.stylesheet", "edge { shape:line;fill-color:#222;arrow-size:3px,2px;}");
        graph.display();
        graph.addAttribute("ui.screenshot", "/Users/ira/Downloads/graph/"+name+"_tpn.jpg");
```

```
public void graphEventCMD(List<Event> emps, List<String> date, List<String> cmd, String name)
throws NullPointerException, EdgeRejectedException, IdAlreadyInUseException, FileNotFoundException, IOException {
    Graph graph = new MultiGraph("Graph2");
    graph.addAttribute("ui.quality");
    graph.addAttribute("ui.antialias");
    Viewer viewer = new Viewer(graph, Viewer.ThreadingModel.GRAPH_IN_ANOTHER_THREAD);
    ViewPanel viewPanel = viewer.addDefaultView(false);
    viewPanel.getCamera().setViewPercent(0.25);
    viewPanel.getCamera().setViewCenter(0, 0, 0);
    for (int i = 0; i < date.size(); i++){
        graph.addNode(date.get(i));
    }
    for (int i = 0; i < cmd.size(); i++){
        graph.addNode(cmd.get(i));
    }
    for (int i = 0; i < date.size(); i++){
        for (int j = 0; j < emps.size(); j++){
            if (distance(date.get(i), emps.get(j).getDate())==0){
                if (graph.getEdge("connected "+emps.get(j).getDate()+emps.get(j).getDC1())==null){
                    graph.addEdge("connected "+emps.get(j).getDate()+emps.get(j).getDC1(), date.get(i), emps.get(j).getDC1());}
            }
        }
    }
    for (Node node : graph) {
        node.addAttribute("ui.label", node.getId());
    }
    graph.addAttribute("ui.stylesheet", "node {size:3px;fill-color:#777;z-index:0;}");
    graph.addAttribute("ui.stylesheet", "edge { shape:line;fill-color:#222;arrow-size:3px,2px;}");
    graph.display();
    graph.addAttribute("ui.screenshot", "/Users/ira/Downloads/graph/"+name+"_cmd.jpg");
}

public void graphEventTrio(List<Event> emps, List<String> date , List<String> tpn, String name)
throws NullPointerException, EdgeRejectedException, IdAlreadyInUseException, FileNotFoundException, IOException {
    Graph graph = new MultiGraph("Graph2");
    graph.addAttribute("ui.quality");
    graph.addAttribute("ui.antialias");
    Viewer viewer = new Viewer(graph, Viewer.ThreadingModel.GRAPH_IN_ANOTHER_THREAD);
    ViewPanel viewPanel = viewer.addDefaultView(false);
    viewPanel.getCamera().setViewPercent(0.25);
    viewPanel.getCamera().setViewCenter(0, 0, 0);
    for (int i = 0; i < date.size(); i++){
        graph.addNode(date.get(i));
    }
    for (int i = 0; i < date.size(); i++){
        for (int j = 0; j < emps.size(); j++){
            if (distance(date.get(i), emps.get(j).getDate())==0){
                if (graph.getNode(emps.get(j).getDestinationProcessName()+" for date:"+date.get(i))==null){
                    graph.addNode(emps.get(j).getDestinationProcessName()+" for date:"+date.get(i));
                }
                if (graph.getEdge("connected "+emps.get(j).getDate()+emps.get(j).getDestinationProcessName())==null){
                    graph.addEdge("connected "+emps.get(j).getDate()+emps.get(j).getDestinationProcessName(),
                    date.get(i), emps.get(j).getDestinationProcessName()+" for date:"+date.get(i));}
            }
        }

}
for (int k = 0; k < tpn.size(); k++){
    for (int i = 0; i < date.size(); i++){
        for (int j = 0; j < emps.size(); j++){
            if (graph.getNode(emps.get(j).getDestinationProcessName()+" for date:"+emps.get(j).getDate())!=null
            && graph.getEdge("connected "+emps.get(j).getDate()+emps.get(j).getDestinationProcessName())!=null){
                if (graph.getNode(emps.get(j).getDC1()+" for date:"+emps.get(j).getDate())==null){
                graph.addNode(emps.get(j).getDC1()+" for date:"+emps.get(j).getDate());
                    if (graph.getEdge("connected "+emps.get(j).getDestinationProcessName()+emps.get(j).getDC1())==null){
                        graph.addEdge("connected "+emps.get(j).getDestinationProcessName()+emps.get(j).getDC1(),
                        emps.get(j).getDestinationProcessName()+" for date:"+emps.get(j).getDate(),
                        emps.get(j).getDC1()+" for date:"+emps.get(j).getDate());
                    }
                }
            }
        }
    }
}
for (Node node : graph) {
    node.addAttribute("ui.label", node.getId());
}
graph.addAttribute("ui.stylesheet", "node {size:3px;fill-color:#777;z-index:0;}");
graph.addAttribute("ui.stylesheet", "edge { shape:line;fill-color:#222;arrow-size:3px,2px;}");
graph.display();
graph.addAttribute("ui.screenshot", "/Users/ira/Downloads/graph/"+name+"_trio.jpg");
```

Chart.java

```
package fuzzystringsearch;

import java.awt.Color;
import java.awt.Font;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;
import org.knowm.xchart.BubbleChart;
import org.knowm.xchart.BubbleChartBuilder;
import org.knowm.xchart.BubbleSeries;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;

/**
 *
 * @author ira
 */
public class Chart {

  public XYChart getChart(List<Event> emps, List<Techniques> all, String title) throws ParseException {
    XYChart chart = new XYChartBuilder().width(800).height(600).title(title).xAxisTitle("date").yAxisTitle("stage").build();
    List<TechniqueChart> ch = new ArrayList();
    chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor(ChartColor.LIGHT_GREY));
    chart.getStyler().setPlotGridLinesColor(Color.BLACK);
    chart.getStyler().setChartBackgroundColor(Color.WHITE);
    chart.getStyler().setLegendBackgroundColor(Color.WHITE);
    chart.getStyler().setChartFontColor(Color.BLACK);
    chart.getStyler().setChartTitleBoxBackgroundColor(Color.WHITE);
    chart.getStyler().setChartTitleBoxVisible(true);

    chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
    chart.getStyler().setPlotGridLinesVisible(false);
    chart.getStyler().setAxisTickPadding(20);
    chart.getStyler().setAxisTickMarkLength(15);
    chart.getStyler().setPlotMargin(20);
    chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.BOLD, 24));
    chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
    chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
    chart.getStyler().setLegendSeriesLineLength(12);
    chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC, 18));
    chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.PLAIN, 11));
    chart.getStyler().setDatePattern("dd-MMM");
    chart.getStyler().setDecimalPattern("#0.000");
    chart.getStyler().setLocale(Locale.UK);

    DateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
    for(Event ev : emps){
        for(Techniques ev2 : all){
            if (ev.getName().contains(ev2.getIdTec())){
            TechniqueChart event = new TechniqueChart();
            event.setId(ev2.getId());
            event.setDate(ev.getDate());
            event.setName(ev2.getIdTec());
            event.setConcat(ev.getDate()+" / "+ev2.getIdTec());
            event.setiddate(ev2.getIdTec()+" / "+ev2.getId()+" / "+ev.getDate());
                if(!ch.contains(event)){
                    ch.add(event);
                }
            }
        }
    }
```

```
  List<Date> xData1 = new ArrayList<Date>();
  List<Integer> yData1 = new ArrayList<Integer>();
  SimpleDateFormat format = new SimpleDateFormat("MMMM d, yyyy", Locale.ENGLISH);
  for(TechniqueChart ev : ch){
      yData1.add(ev.getId());
      Date   date1 = format.parse (ev.getDate());
      xData1.add(date1);
  }
  XYSeries series = chart.addSeries("Techniques", xData1, yData1);
  series.setLineColor(XChartSeriesColors.LIGHT_GREY);
  series.setMarkerColor(Color.BLACK);
  series.setMarker(SeriesMarkers.CIRCLE);
  series.setLineStyle(SeriesLines.SOLID);
  new SwingWrapper(chart).displayChart();
  return chart;
}

public XYChart getFinalChart(List<Event> emps, String title) throws ParseException, FileNotFoundException, IOException {
  XYChart chart = new XYChartBuilder().width(800).height(600).title(title).xAxisTitle("date").yAxisTitle("stage").build();
  List<TechniqueChart> ch = new ArrayList();
  chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor(ChartColor.LIGHT_GREY));
  chart.getStyler().setPlotGridLinesColor(Color.BLACK);
  chart.getStyler().setChartBackgroundColor(Color.WHITE);

  chart.getStyler().setLegendBackgroundColor(Color.WHITE);
  chart.getStyler().setChartFontColor(Color.BLACK);
  chart.getStyler().setChartTitleBoxBackgroundColor(Color.WHITE);
  chart.getStyler().setChartTitleBoxVisible(true);
  chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
  chart.getStyler().setPlotGridLinesVisible(false);
  chart.getStyler().setAxisTickPadding(20);
  chart.getStyler().setAxisTickMarkLength(15);
  chart.getStyler().setPlotMargin(20);
  chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.BOLD, 24));
  chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
  chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
  chart.getStyler().setLegendSeriesLineLength(12);
  chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC, 18));
  chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.PLAIN, 11));
  chart.getStyler().setDatePattern("dd-MMM");
  chart.getStyler().setDecimalPattern("#0.000");
  chart.getStyler().setLocale(Locale.UK);
  List<String> list = new ArrayList();
  CSVFormat format = CSVFormat.EXCEL.withHeader().withDelimiter(';');
  CSVParser parser = new CSVParser(new FileReader("/Users/ira/Downloads/tecs.csv"), format);
  DateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
  for(CSVRecord record : parser){
      TechniqueChart event = new TechniqueChart();
      event.setId(Integer.valueOf(record.get(0)));
      event.setDate(record.get(2));
      event.setName(record.get(1));
      event.setConcat(record.get(2)+" / "+record.get(1));
      event.setiddate("none");
      ch.add(event);
  }
  List<Date> xData1 = new ArrayList<Date>();
  List<Integer> yData1 = new ArrayList<Integer>();
  SimpleDateFormat format1 = new SimpleDateFormat("MMMM d, yyyy", Locale.ENGLISH);
  for(TechniqueChart ev : ch){
      yData1.add(ev.getId());
      Date   date1 = format1.parse (ev.getDate());
      xData1.add(date1);
  }
  XYSeries series = chart.addSeries("Techniques", xData1, yData1);
  series.setLineColor(XChartSeriesColors.LIGHT_GREY);
  series.setMarkerColor(Color.BLACK);
  series.setMarker(SeriesMarkers.CIRCLE);
  series.setLineStyle(SeriesLines.SOLID);
  new SwingWrapper(chart).displayChart();
  return chart;
}


public BubbleChart getBubbleChart(List<Event> emps, List<Techniques> all, List<TechniqueChart> ch, String title) throws ParseException {
  BubbleChart chart = new BubbleChartBuilder().width(800).height(600).title(title).xAxisTitle("date").yAxisTitle("tactic stage").build();
  DateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
  for(Event ev : emps){
      for(Techniques ev2 : all){
```

```
            if (ev.getName().contains(ev2.getIdTec())){
                TechniqueChart event = new TechniqueChart();
                event.setId(ev2.getId());
                event.setDate(ev.getDate());
                event.setName(ev2.getIdTec());
                event.setConcat(ev.getDate()+" / "+ev2.getIdTec());
                event.setiddate(ev2.getIdTec()+" / "+ev2.getId()+" / "+ev.getDate());
                ch.add(event);
            }
        }
    }

    Map<String, Integer> tec = new HashMap<>();
    for (TechniqueChart i : ch) {
        Integer j = tec.get(i.getiddate());
        if (j==null)
            j=0;
        tec.put(i.getiddate(), j+1);
    }

    Map<TechniqueChart, Integer> x = new HashMap<>();
    tec.entrySet().forEach((Map.Entry<String, Integer> entry) -> {
        for (TechniqueChart i : ch) {
            if (i.getiddate().equals(entry.getKey())){
                x.put(i, entry.getValue());
            }
        }
    });
    int sum = x.values().stream().mapToInt(Integer::intValue).sum();
    List<Integer> xData1 = new ArrayList<Integer>();
    List<Integer> yData1 = new ArrayList<Integer>();
    List<Integer> bubble = new ArrayList<Integer>();
    SimpleDateFormat format = new SimpleDateFormat("MMMM d, yyyy", Locale.ENGLISH);
    SimpleDateFormat format1 = new SimpleDateFormat("yyyyMMdd", Locale.ENGLISH);
    x.entrySet().forEach((entry) -> {
        try {
            Date date1 = format.parse (entry.getKey().getDate());
            String date2 = format1.format(date1);
            xData1.add(Integer.valueOf(date2));
            yData1.add(entry.getKey().getId()*100);
            bubble.add(entry.getValue());
        } catch (ParseException ex) {
            Logger.getLogger(Chart.class.getName()).log(Level.SEVERE, null, ex);
        }
    });

    BubbleSeries series = chart.addSeries("Techniques", xData1, yData1, bubble);
    new SwingWrapper<BubbleChart>(chart).displayChart();
    return chart;
    }

}
```