МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Комплексна система захисту інформації в локальній мережі. Моніторинг»

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

Керівник роботи — **Симоновський Ю.В.**

Студента групи КБ – 61 — **Макаров А.М.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 г.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи КБ-61 спеціальності "Кібербезпека" денної форми навчання Макарова Андрія Миколайовича.

Тема: "**Комплексна система захисту інформації в локальній мережі. Моніторинг.**"

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів захисту мережі від загроз інформаційної безпеки; 2) постановка завдання й формування завдань дослідження; 3) характеристика експертних систем для розв'язування задач інформаційної безпеки; 5) розробка інформаційного й програмного забезпечення SIEM системи; 6) тестування та аналіз результатів.

Дата видачі завдання "_____"_____2020 р.

Керівник випускної роботи _____ Симоновський Ю.В.

Завдання прийняв до виконання _____ Макаров А.М.

РЕФЕРАТ

Записка: 42 сторінок, 10 рисунків, 1 додаток, 15 джерел.

Об'єкт дослідження – слабоформалізований процес моніторингу безпеки мережі.

Мета роботи – розробка системи моніторингу безпеки мережі.

Результати – сформовано вхідний математичний опис SIEM системи для моніторингу безпеки мережі; розроблено основні компоненти SIEM системи, для моніторингу; правил; працездатність системи перевірена на прикладі реалізації сценарію “перебір паролю”.

МОНІТОРИНГ, УПРАВЛІННЯ БЕЗПЕКОЮ, КОМПЮТЕРНА МЕРЕЖА, ІНЦИНДЕНТ БЕЗПЕКИ, SIEM СИСТЕМА

ЗМІСТ

Вступ

Захист мережі на сьогодні потребує використання різноманітних засобів забезпечення безпеки, наприклад: комплекс виявлення атак на мережу, захист від спаму, антивіруси, міжмережеві екрани, різноманітні програми сканування безпеки і тому подібні. З цього слідує збільшення кількості апаратних і програмних засобів захисту мережі, що в свою чергу значимо збільшує кількість вхідної інформації, яку необхідно аналізувати для забезпечення безпеки. Адміністратори, які відповідають за мережу, витрачають велику кількість ресурсів для виконання одноманітної роботи, що значно знижує ефективність в цілому.

Отже, зараз актуально використовувати нові ефективні методи для моніторингу, та аналізу подій в мережах.

1 АНАЛІЗ ПРОБЛЕМИ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналітичний огляд методів захисту мережі від загроз інформаційної безпеки

Під загрозою з точки зору безпеки слід розуміти сукупність умов і факторів, які потенційно призводять до порушення функціонування комп'ютерної мережі в цілому, в тому числі - контрольованим мережею активів (даними). Загрози зазвичай поділяються на: умисні (усвідомлене заподіяння шкоди) і природні. До перших відносяться несанкціоновані підключення, виток інформації, порушення функціонування мережі тощо, до других - форс-мажорні обставини та нещасні випадки, а також помилки внаслідок збоїв апаратури. При цьому велика кількість вразливостей безпеки існують через людський фактор – недостатня компетенція користувача, або повне недотримання інструкцій (наприклад, недотримання парольної політики)[1]. При аналізі комп'ютерної мережі потрібно взяти до уваги всі типи загроз та вразливостей. Загрози, які мають природний характер, досить легко описуються в плані ризиків а захищатися від них не вимагає глибокого аналізу. Але загрози причина яких є людський фактор, потребують особливої уваги через, неможливість передбачити напевне дії людини, навіть якщо вона немає мотиву, та намірів реалізувати загрозу. З іншого боку, людський фактор має значення в тих випадках, коли дисфункція системи якимось чином загрожує людині, що особливо важливо з футуристичної точки зору - час, коли робототехніка стане звичною в побуті, відноситься до близького майбутньому [2].

У будь-якої комп'ютерної мережі навіть за добу відбуваються десятки і сотні тисяч подій, що мають відношення до інформаційної безпеки, при цьому переважна більшість - це нормальне функціонування мережі, і лише деякі з них є сигналами про потенційних або ж фактичних порушеннях безпеки. Таким чином, вкрай актуальним завданням є моніторинг стану мережі, який

дозволяє відстежувати не тільки поточний стан мережі, а й зміни в ній як в динамічній системі. В цілому, моніторинг комп'ютерної мережі можна визначити як постійне спостереження за об'єктами і факторами, що впливають на функціонування мережі, а також як аналіз результатів спостереження, включаючи зберігання і узагальнення відповідної інформації [3].

Для знаходження загроз адміністратор використовує велику кількість інструментів: системи аналізу мережевого трафіку, різноманітні антивіруси, брандмауери, системи для забезпечення криптографічної безпеки, програми для виявлення різних видів атак, наприклад IDS. Попри таку різноманітність, ці засоби не використаються постійно, а тільки під час уже реалізованої загрози, і тільки інколи для запобігання. Існуючи методи, для аналізу націлені на знаходження популярних, і всім відомим загроз, та вразливостей, що не дає змоги для виявлення маловідомих, або модифікованих загроз.

Для здійснення моніторингу потрібна наявність в комп'ютерній мережі єдиної системи управління всіма вузлами і сегментами мережі, яка буде служити базою для розгортання системи моніторингу, що дозволить здійснювати динамічний контроль стану всіх важливих вузлів і елементів мережі в реальному часі, а також накопичувати відповідну статистику для подальшого використання в прогнозуванні ситуацій порушення функціональності мережі. Наявність підсистеми моніторингу має враховуватися ще на етапі проектування мережі, оскільки вимагає виділених ресурсів [4].

У загальному вигляді система моніторингу повинна складатися з наступних компонентів:

• підсистема збору інформації, що надходить від усіх наявних засобів захисту;

• сервер подій, призначений для централізованої обробки інформації, що надходить про події, що мають відношення до безпеки мережі, відповідно до правил, заданими адміністратором мережі;

• сервер зберігання даних - як первинної інформації (подій), як і результатів аналізу;

• призначений для користувача інтерфейс управління системою моніторингу, що дозволяє здійснювати контроль і управління в реальному масштабі часу.

Збір інформації, необхідної для контролю стану мережі, може бути представлений як:

• аналіз мережевих пакетів (включаючи декодування в разі необхідності);

• моніторинг функціонування активного мережного обладнання;

• моніторинг робочого стану кабельної системи, а також бездротових з'єднань;

• моніторинг функціонування серверів і робочих станцій (останні можуть поділятися за ступенем важливості з точки зору безпеки);

• моніторинг роботи операційних систем і програмних додатків.

Отже, крім безпосередніх даних від засобів моніторингу, необхідно зберігати і інші дані, що мають відношення до безпеки мережі: відомості про виконання політик інформаційної безпеки, про уразливість кінцевих систем, збої працездатності, методи відновлення працездатності системи і т.д. Така сукупність інформації може бути надалі використана для прийняття рішень щодо забезпечення безпеки мережі, тому зберігання цих даних необхідно. Однак об’єм інформації, та їх різна природа, ускладнюють моніторинг параметрів мережі. Необхідно створення методології, яка дозволить робити характеристику та аналіз необхідної інформації, в реальному часі.

Першим методом, для рішення поставленої задачі є моментальний аналіз наявної бази даних, та обчислення оцінки, з різних джерел, результату збору інформації за заздалегідь визначеним параметрам. Цей метод дозволяє об’єднати багато різних типів інформації, що моніториться системою, до зрозумілих для адміністратора показників, що потім зможуть бути використанні під час вибору правильного плану дій, людиною яка відповідає за безпеку мережі. В дослідженнях показано, що використання такого метода

під час проведення моніторингу у локальній мережі , зможе захистити від загроз приблизно на 30% атак більше. [5].

В сучасному світі, не можна вже дивитися на безпеку в інформаційних системах, як на підтримку завідома відомого оптимального стану. Потрібний підхід для підтримки захисту, який зможе забезпечити необхідний рівень захисту, але при цьому без відомого наперед одного сценарію подій, а з певної кількості таких сценаріїв, які можуть комбінуватися, та відповідати сучасним стандартам безпеки[6].

Але існування оцінених необхідних параметрів з різних джерел назавжди є достатнім для підтримки системи в безпеці, якщо це велика та складна комп'ютерна мережа. Вирішення управлінських питань в таких випадках залежить від багатьох критеріїв та повинне здійсненне враховуючи багато обмежень. В результаті людина, яка приймає рішення, часто робить рішення, яке засноване на простій методології аналізу, головна мета яких швидкість всупереч якості. Вона застосовує прості, але давно застарілі методології. Отже необхідно мати в інформаційній системі інструмент для вирішення питань безпеки. Вона могла би виконувати всю одноманітну роботу, замість адміністратора. Зараз гарним рішенням є система, яка знає рішення, якщо воно єдине, воно яке виконуються, та сповіщається адміністратора, але якщо є декілька варіантів рішення, то вибір робить компетентна людина.

У загальному вигляді найбільш оптимальним для управління є напівавтоматичний режим: якщо є єдине відоме рішення проблеми, то воно застосовується автоматично (додатково доцільно повідомляти фахівця з безпеки), а в разі наявного вибору варіантів дії, а також при відсутності відомих рішень проблеми, рішення має приймати фахівець [7].

Функціонування системи підтримки прийняття рішень вимагає ведення бази даних, в якій повинні фіксуватися наступні параметри:

• опис інциденту, включаючи класифікацію такого згідно з розробленою системою;

• наявні тенденції подальшого поширення збою системи (локалізація події);

• список активів за категоріями, порушених інцидентом;

• список порушених збоєм мережі користувачів з урахуванням їх рівнів доступу і т.д .;

• вплив збою на функціонування і ресурси мережі в цілому;

• наявність в базі алгоритмів протидії загрозі.

Через складність поставленої проблеми, в якості рішення можна використовувати нейронні мережі для проведення обробки інформації та інцидентів пов’язаних з моніторингом. Їх здатність до самонавчатися дає їм великі переваги: збій в інформаційних системах, та загрози з вразливостями, можуть бути представлені як нелінійні залежності, аргументи якого не завжди пов'язані один з одним, аналітично це здійснити складно. Один з варіантів - це застосування мережі Гопфілда[8].

Типові стандартні рішення безпеки рідко передбачають механізм адаптації до різних конфігурацій мереж, внаслідок чого виключається можливість оптимального використання вільних ресурсів мережі. Більш того, часто відбувається конкуренція за обчислювальні ресурси з додатками, що працюють в мережі в той же момент часу. Для оптимізації роботи потрібно поділ інформаційних потоків в реальному часі, а також скорочення часу опитування робочих станцій, здійснюваного системою моніторингу. [8].

У найбільш загальному вигляді система моніторингу повинна здійснювати два стратегічних видів контролю.

1. Контроль цілісності системи, тобто стану мережі, при якому комп'ютерна система функціонує як логічно єдина система апаратних і програмних засобів (елементів системи), повноцінно забезпечують роботу захисних механізмів, включаючи логічну коректність роботи і нормальне функціонування в плані нейтралізації загроз безпеці. Також сучасні системи

контролю цілісності зобов'язані відстежувати розподілені конфігурації мережі і мати захист від несанкціонованої модифікації потоків даних між вузлами мережі.

2. Контроль захищеності системи, тобто спроби санкціонованого «злому» інформаційної системи, яка здійснюється організацією-власником з метою виявлення наявних вразливостей в захисті мережі, які доцільно виявляти раніше зловмисників. Особливо корисний цей метод при введенні нового програмного забезпечення (поновлення істотно відрізняються версій), а також в разі зміни кадрового складу співробітників, що працюють з відповідним вузлом мережі. Системи моніторингу комп'ютерних мереж є необхідною складовою загального забезпечення інформаційної безпеки організації. Ефективність забезпечення інформаційної безпеки залежить від того, наскільки однозначно сформульовані вимоги до оперативних (поточних), тактичним і стратегічним завданням, і наскільки цілісно при цьому забезпечено їх взаємозв'язок. Системи моніторингу як частина системи інформаційної безпеки призначені для забезпечення вирішення оперативних і частково тактичних завдань, сприяючи досягненню стратегічних цілей безпеки.

1.2 Постановка задачі

Наведений вище огляд методів забезпечення безпеки мережі вказує на перспективність створення системи для моніторинга мереж. У дипломній роботі необхідно розробити систему моніторинга мережі, та управління подіми безпеки за допомогою системи SIEM.

При цьому необхідно виконати такі завдання:

1) описати методологію розробки SIEM;
1) описати, та організувати тестове оточення;
3) розробити колектор SIEM системи;
4) розробити ядро кореляції SIEM системи;
5) перевірити ефективність розробленої системи.

2 ДОСЛІДЖЕННЯ УПРАВЛІННЯ ПОДІЯМІ БЕЗПЕКИ ЗА ДОПОМОГОЮ SIEM СИСТЕМ

Розвиток та різноманітні процеси, в локальних мережах, вказують на збільшення інтенсивності обміну в інформаційних системах. В даний час актуальною проблемою є проблема виявлення та локалізації тимчасових відомостей, що знаходяться всередині мереж. Також є велика кількість інформаційних атак націлені на ресурси інформаційних систем.

Способи розповсюдження шкідливої інформації в локальних інформаційних системах також модифікуються, що робить процеси виявлення, більш трудомісткім та вимагає застосування різних засобів захисту інформації. Інтенсивність інформаційних подій в мережах може досягати декількох мільйонів в день, тому виникає проблема знаходження і локалізації шкідливої інформації у величезних інформаційних масивах. При цьому обробка подібних подій в ручному режимі є неможливою, оскільки необхідно значних людських і часових затрат, а також неприпустимих з точки зору ефективності апаратно-програмних ресурсів [9].

Для побудови ефективної, в тому числі і з економічної точки зору, системи захисту інформації необхідно визначити і проаналізувати основні проблеми захисту інформації конкретної мережі.

В ході аналізу проблеми захисту інформації на об'єктах інформатизації, на підставі особливостей кожного об'єкта, необхідно визначити інформаційні ресурси, які використовуються в даній інформаційній системі. До них, як правило, відносяться:

- файлові сервіси;
- локальні обчислювальні мережі;
- сервери баз даних;
- сервери, запропоновані;
- офісні додатки;

- система віртуалізації.

Також необхідно визначити категорії доступності інформації, що обробляється в даній інформаційній системі. Після цього складається перелік основних технічних засобів і систем, за допомогою яких відбувається обробка інформації обмеженого доступу.

Аналіз проблем безпеки інформації передбачає визначення основних факторів, що впливають на організацію комплексної системи захисту інформації. Перш за все до них відносяться форма власності підприємства, сфера основних видів діяльності, особливості територіального розташування підприємства, ступінь автоматизації основних процедур обробки інформації, що захищається і ряд інших факторів [10].

На підставі істотних факторів, об'єкта захисту інформації, аналізу та оцінки загроз, як правило, формується необхідний перелік способів і засобів захисту інформації. При цьому інтенсивність інформаційного обміну, складність сучасних алгоритмів обробки інформації, різноманітність загроз і засобів захисту від них обумовлюють необхідність застосування різноманітних автоматизованих засобів і систем для вирішення завдань забезпечення інформаційної безпеки.

Одним із сучасних підходів до вирішення завдань захисту інформації мереж є впровадження SIEM-технології.

2.1 Моніторинг подій інформаційної системи. SIEM.

SIEM (Security information event management) - клас систем забезпечення інформаційної безпеки, що з'явилися в результаті злиття SEM-систем і SIM-систем. Основною функціональною відмінністю даних систем є те, що SEM-системи призначені для аналізу інформації в режимі реального часу, а SIM системи аналізують вже накопичену інформацію [11].

Основною функцією SIEM-систем є аналіз інформації, що надходить від різних джерел, таких як системи DLP, засоби антивірусного захисту інформації, міжмережеві екрани, системи обліку трафіку, сканери уразливості

і т.д. [12]. На основі аналізу даних з цих джерел виявляються відхилення від нормального функціонування, заданого критеріями безпеки, і в разі виявлення відбувається оповіщення адміністратора безпеки. Крім того, типова SIEM-система може використовуватися для:

- аналіз інформації, що знаходиться з різних джерел;
- подання доказової бази при розслідуванні інцидентів інформаційної безпеки;
- подання розробленої інформації, необхідної для аудіоінформаційної безпеки;
- забезпечення постійної роботи сервісних служб шляхом роботи в їх роботі;
- розробка інформаційно-телекомунікаційної системи.

Типові функціональні можливості SIEM-систем припускають виявлення наступних подій та інцидентів інформаційної безпеки:

- мережевих атак у внутрішньому і зовнішньому периметрах;
- вірусних епідемій або окремих вірусних заражень;
- спроб несанкціонованого доступу до конфіденційної інформації;
- шахрайства;
- помилок і збоїв в роботі інформаційних систем;
- вразливостей різної природи;
- помилок конфігурацій в засобах захисту та інформаційних системах;
- цільових атак.

Основними завданнями забезпечення інформаційної безпеки, які ставляться перед SIEM-системою, як правило, є наступні:

- централізоване зберігання журналів подій;
- обробка та кореляція подій;
- оповіщення про інциденти;
- розслідування інцидентів;
- управління інцидентами (інцидент-менеджмент).

2.2 Основні компоненти SIEM систем

Типове рішення SIEM-системи включає в себе кілька функціональних компонентів, які зображені на рисунку 2.1:

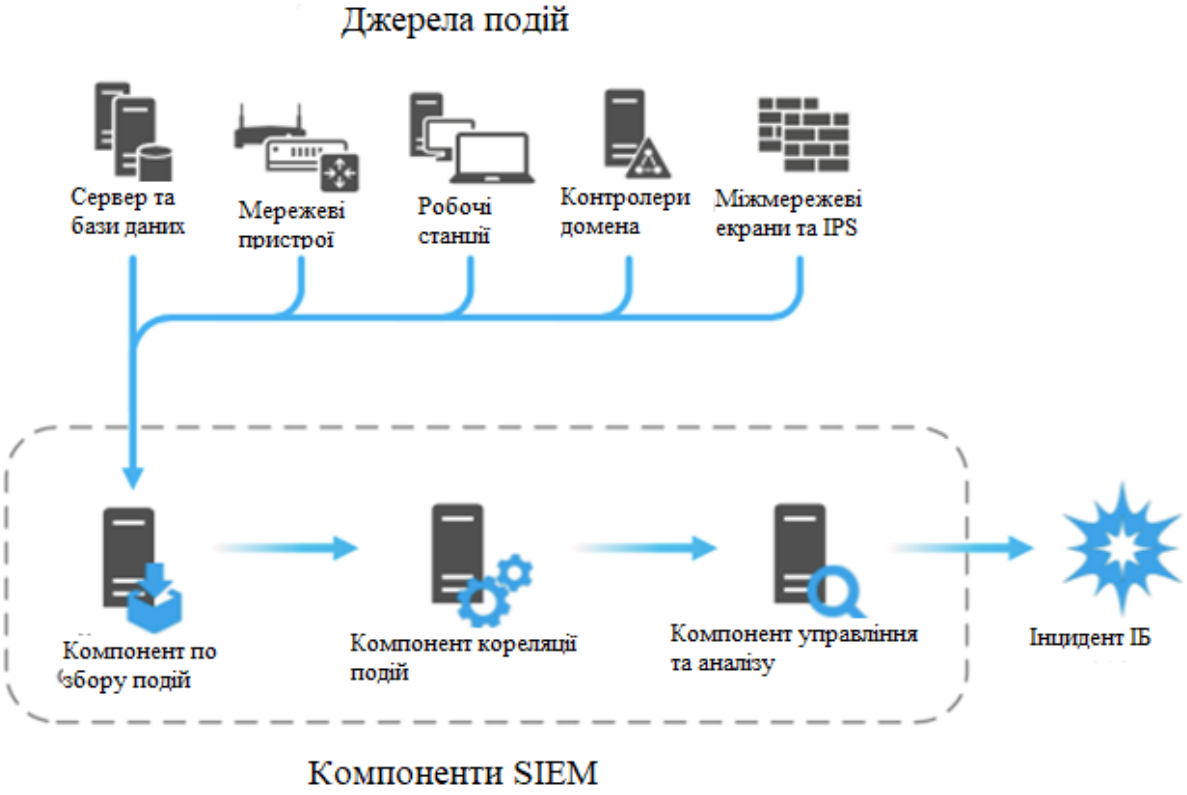

Рисунок 2.1 Основні компоненти SIEM

- агенти, що встановлюються на інформаційну систему, яка моніториться (актуально для операційних систем; агент являє собою резидентну програму (сервіс, демон, служба), яка локально збирає журнали подій і по можливості передає їх на сервер);
- колектори на агентах, які, по суті, являють собою модулі (бібліотеки) для розуміння конкретного журналу подій або системи;
- сервери-колектори, призначені для попередньої акумуляції подій від безлічі джерел;
- сервер-коррелятор, що відповідає за збір інформації від колекторів і агентів і обробку за правилами і алгоритмами кореляції;
- сервер баз даних і сховища, який відповідає за зберігання журналів подій.

Функціонування SIEM-системи доцільно деталізувати на рівні:

- збір лог-файлів і формування необхідних даних від різних джерел;

- нормалізація даних, яка полягає у приведенні подій з однаковим змістом до загального формату;
- кореляція подій системи, важливих для забезпечення безпеки, шляхом знаходження зв'язків між ними, наприклад, підбір паролів, зараження шкідливим кодом, аномальна активність в системі, зміна критичних параметрів системи і т.п .;
- організація зберігання лог-файлів;
- реагування на інциденти, в тому числі повідомлення про важливі події для інформаційної безпеки;
- візуалізація інцидентів, формування звітних документів.

Дана структура є спільною для всіх інформаційних систем. При цьому для побудови ефективної системи захисту інформації необхідно враховувати особливості конкретної мережі. Впровадження будь-якого SIEM-рішення з урахуванням особливостей інформаційної системи передбачає розробку структурно-функціональної моделі даної системи. Структурно-функціональна модель системи захисту інформації (СЗІ) включає в себе перелік структурних компонентів обладнання, а також їх функціональні зв'язки і можливості при вирішенні завдання аналізу і захисту інформації (рис. 2).

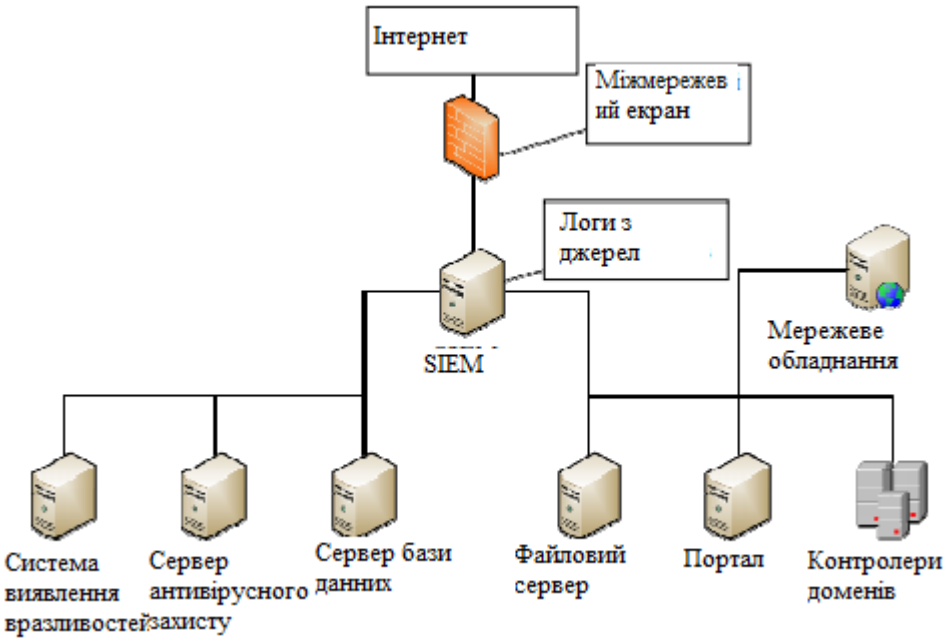

Рисунок 2.2 – Структурно функціональна модель SIEM-системи

До типових структурних компонентів відносяться:

- міжмережевий екран;
- поштові послуги;
- бази даних;
- система виявлення вразливостей;
- антивірусний захист;
- локальний портал;
- файловий сервер.

За функціональним можливостям структурні компоненти діляться на кілька груп:

- інструменти, скануючі мережу в пошуках вразливостей;
- інші засоби;
- міжмережевий екран, що обмежує переміщення пакетів за межі локальної мережі;
- пристрої, які надсилають технології в SIEM-системі.

Побудова структурно-функціональної моделі з урахуванням особливостей певної мережі дозволяє оптимізувати функціональні можливості SIEM-системи і, як наслідок, побудувати більш ефективну систему захисту інформації. Функціональні можливості SIEM-системи забезпечуються визначенням необхідних і достатніх для досягнення цілей її роботи апаратних ресурсів. Апаратні ресурси, необхідні для забезпечення стабільної роботи SIEM-системи, можна розділити на дві основні групи: ресурси для апаратної частини; ресурси для зберігання логів. Ресурсами для апаратної частини є: серверна станція; пристрій ОЗУ; процесори; жорсткий диск для роботи операційної системи. Належної рівень відмовостійкості системи забезпечується застосуванням RAID / SSD-технологій для зберігання інформації на жорстких дисках.

Розрахунок необхідного обсягу сховища логів визначається за формулою [13]:

$$Fi = NV_i \quad (2)$$

де F_i - обсяг надходять подій з секунду; N - максимальне число подій в секунду;

V_i - середній розмір однієї події, поміщеного в систему зберігання.

Крім того, необхідно врахувати додатковий обсяг пам'яті для стабільного швидкодії і зарезервувати додатково 20% місця дискового простору. Період часу для збереження логів, як правило, становить 90 днів, що можна порівняти з періодом часу, необхідного для складання квартального звіту по роботі системи. При цьому час зберігання інцидентів може залежати від ступеня їх критичності. Отже, формула знаходження обсягу сховища, необхідного для зберігання логів, матиме вигляд:

$$F = 1.2 \sum_{i=1}^{T} F_i \quad (1)$$

де F - необхідний простір для зберігання логів; F_i - обсяг надходять подій в секунду; T - період зберігання логів.

Після установки SIEM-системи необхідно підключити джерела подій інформаційної безпеки і написати правила обробки даних подій. Написання правила є послідовність наступних дій:

- імітація інциденту;
- знаходження записи про цю подію в логах джерела;
- визначення унікальних атрибутів в коді даної події;
- при необхідності виділення з коду будь-яких атрибутів в поле події, використовуючи регулярні вирази (створення парсеров);
- створення правила в редакторі правил QRadar з урахуванням унікальних атрибутів події;
- повторна імітація інциденту для перевірки працездатності правила.

Схема алгоритму написання правила представлена на рис. 2.3.

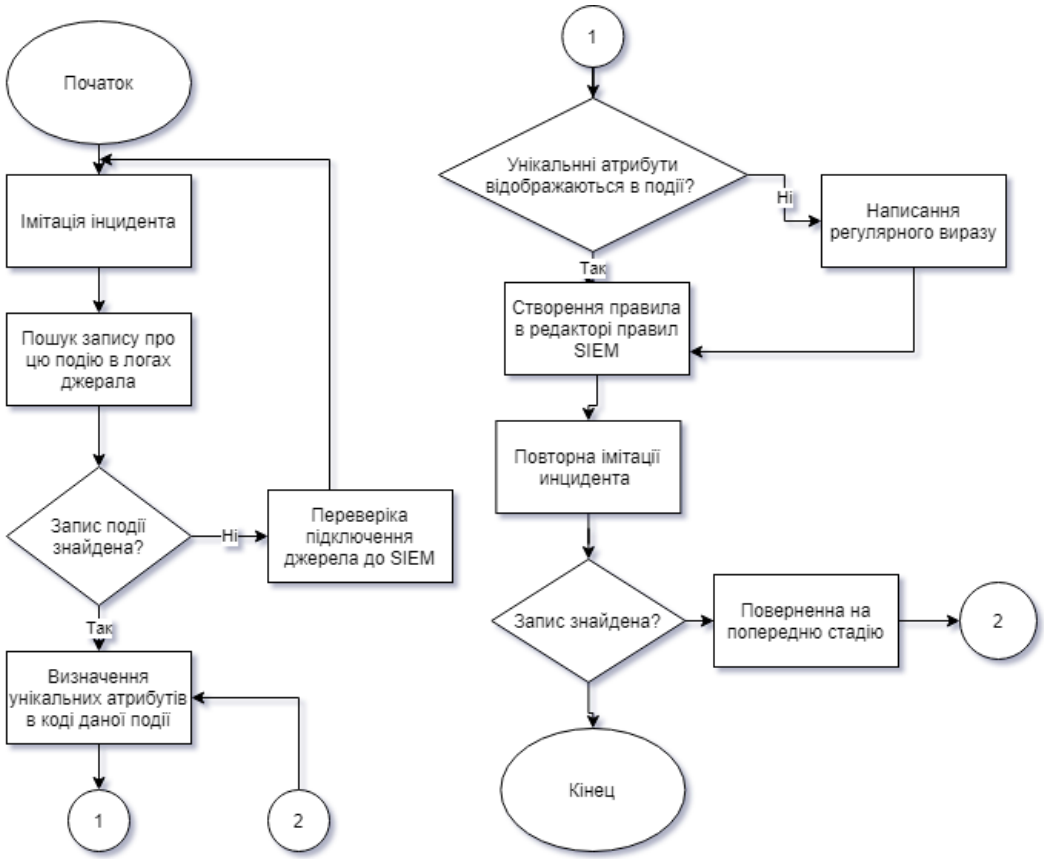

Рисунок 2.2 – Алгоритм створення правила

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Програмні продукти для проектування SIEM системи

Згідно з дослідженням Garther, в число лідерів увійшли наступні системи [14]: Splunk, IBM і LogRhythm. Їх коротка характеристика:

Компанія IBM пропонує комплексне рішення в області SIEM-систем, яке називається Tivoli Security Information and Event Manager (TSIEM). TSIEM дозволяє, з одного боку, проводити аудит подій безпеки на відповідність внутрішнім політикам і різним міжнародним стандартам, а з іншого боку - здійснювати обробку інцидентів, пов'язаних з інформаційною безпекою, і виявляти атаки та інші загрози для елементів інфраструктури. В області подання та зберігання подій TSIEM використовує запатентовану методику W7 (Who, did What, When, Where, Wherefrom, Where to and on What), відповідно до якої всі події трансформуються в єдиний формат, зрозумілий адміністраторам безпеки, аудиторам і управлінцям. Також TSIEM володіє розвиненими можливостями щодо формування звітів і моніторингу активності користувачів.

Splunk - це ще одне рішення для ведення комерційних журналів подій, яке позиціонується як рішення «Пошук в IT» і вбудовується в такі продукти, як Cisco System IronPort. Завдяки веб-інтерфейсу Splunk інтуїтивно зрозумілий в налаштуванні і управлінні. Splunk використовує досить зручний для користувача підхід до проектування інтерфейсів, спрощуючи початковий досвід для менш досвідченого адміністратора. Як і у багатьох аналогічних продуктів для ведення журналів, можливість створення звітів є частиною базового продукту і, в разі Splunk, вона відносно проста у використанні. Поширені типи форматів представлення даних доступні з розкривних меню на екрані. Одна з приємних сторін веб-інтерфейсу Splunk полягає в тому, що будь-який звіт може бути наданий у вигляді URL-адреси, що дозволяє іншим

людям в організації переглядати конкретні звіти, які системний адміністратор створює для них.

Але всі ці системи платні, тому було прийнято рішення розробити власну SIEM систему, для моніторингу безпеки в локальній мережі.

Як сервера будемо використовувати збірку XAMPP for Windows, версія 7.1.9 (Apache 2.4.27 + PHP 7.1.9).

Для організації сховища даних пропонується використовувати документую базу даних MongoDB. Основна причина такого вибору – це знайомство з підходами NoSQL. Крім того, навіть поверхове вивчення комерційних SIEM систем дозволяє зробити висновок про те, що більшість провідних виробників разом з традиційними SQL базами даних в своїх рішеннях застосовують технології NoSQL / NewSQL.

Для організації обміну даними між компонентами системи, що розробляється будемо використовувати брокер повідомлень RabbitMQ, одне з найпоширеніших рішень подібного класу.

Розробка ядра обробки подій безпеки ведеться на C#. Мова досить проста для розуміння, що узгоджується з освітніми цілями проекту. Використовуємо Visual Studio Community 2017.

3.2 Організація тестового оточення

В якості сервера будемо використовувати збірку XAMPP for Windows, версія 7.1.9 (Apache 2.4.27 + PHP 7.1.9). Завантажуємо і встановлюємо XAMPP за допомогою інсталятора, вибираємо папку для установки за замовчуванням - «c:\xampp\». Для перевірки правильності встановлення веб-сервера досить відкрити панель управління XAMPP Control Panel і запустити модуль Apache. Після чого в браузері за адресою «http://127.0.0.1/» відкриється сторінка проекту XAMPP.

Завантажуємо з офіційного сайту і встановлюємо встановлення пакету MongoDB Community Server версії 3.4.10, за інструкцією з офіційного сайту. Перевіряємо працездатність сервера, створивши в колекції (collection) test

новий документ (document) з полем (field) {a: 1}. Спроба знайти документ в колекції повинна завершитися успіхом.(Рис 1.)

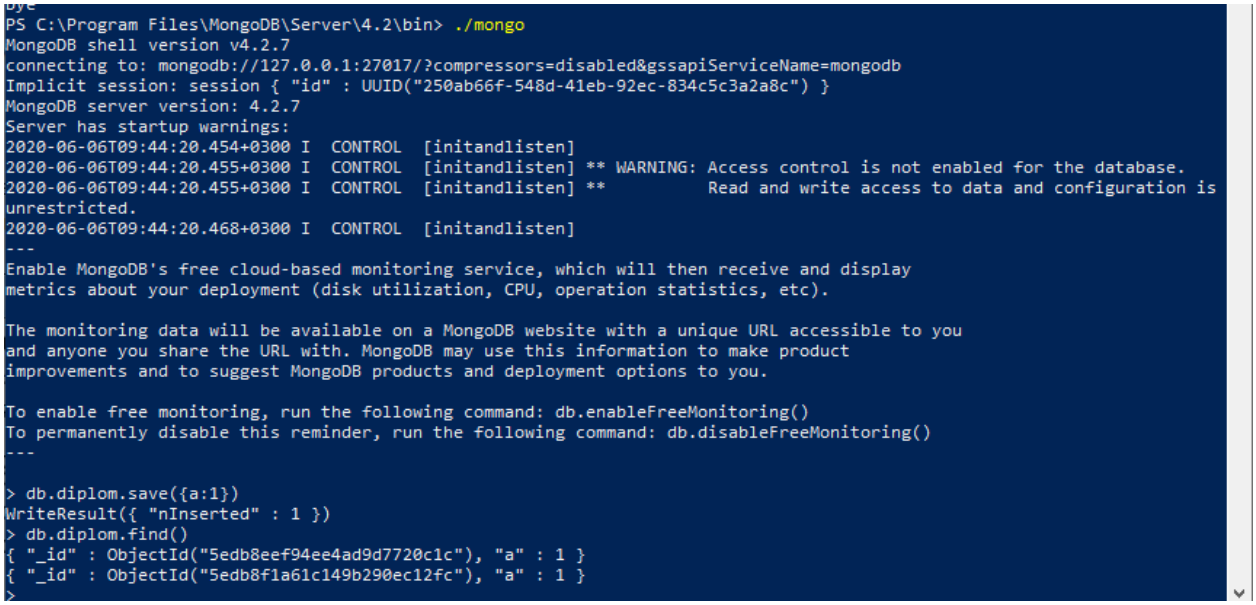

Рисунок 3.1 - Приклад роботи з MongoDB

Встановлення і настройка брокера повідомлень RabbitMQ. Установник налаштує RabbitMQ Server як службу. Для подальших перевірок перейдемо в каталог «c:\ProgramFiles\ RabbitServer\rabbitmq_server-3.8.4\sbin\». Виконуємо команду запуску служби: rabbitmq-service start. Перевіряємо статус командою: rabbitmqctl status Для візуального контролю за функціонуванням брокера повідомлень підключимо відповідний плагін - Management plugin: rabbitmq-plugins enable rabbitmq_management. Після запуску плагіна в браузері за адресою «http://127.0.0.1:15672/» буде доступна панель адміністрування (обліковий запис за замовчуванням - guest / guest). Після перевірки підключення до панелі адміністрування вважаємо встановлення та налаштування брокера повідомлень успішною.

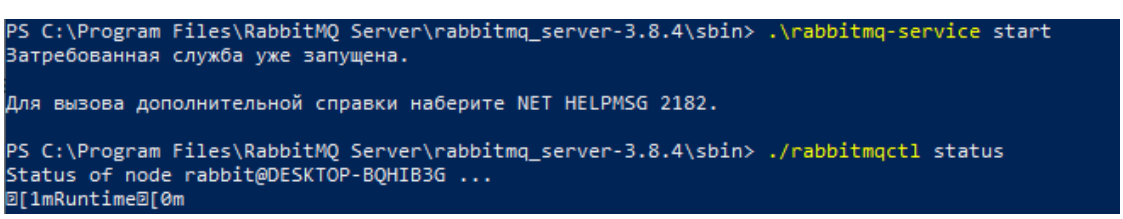

Рисунок 3.1 – Приклад запуску RabbitMQ

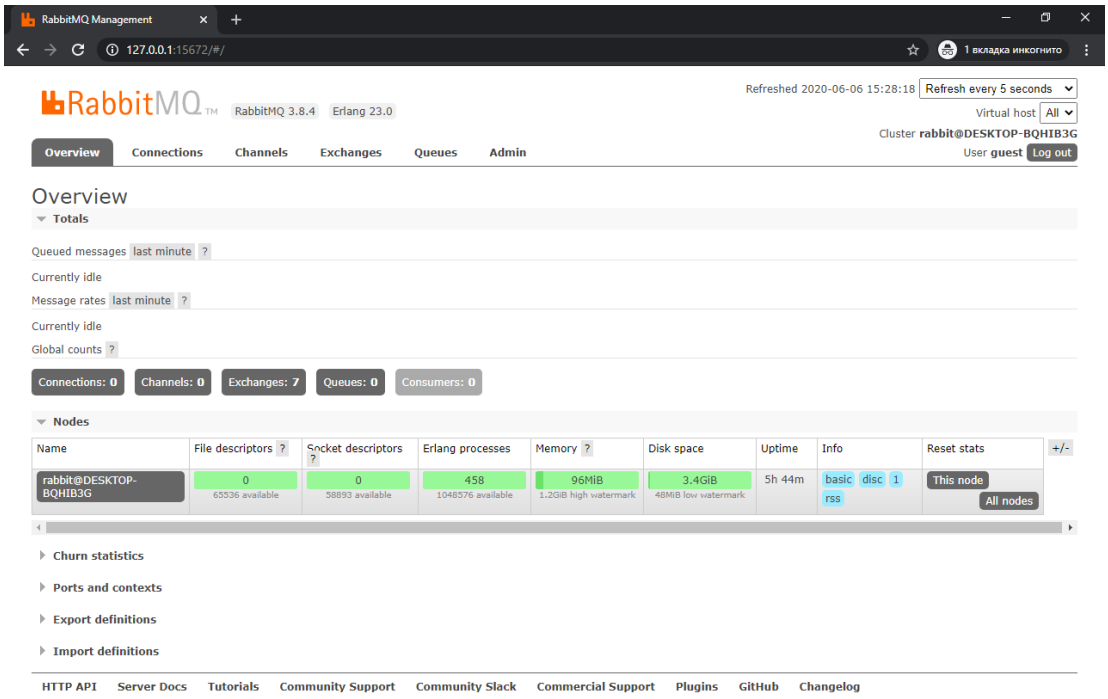

Рисунок 3.3 – Приклад роботи RabbitMQ

В якості елемента для моніторингу буде використаний сервер, в якому припустимо є додаток з доступом до панелі адміністратори постійний моніторинг щоб недопустимий несанкціонований доступ. Для цього було розроблено сторінку, що реалізує функцію аутентифікації користувача. Для реалізації використовуємо мову PHP. Скрипт admin.php запитує ім'я користувача та пароль і порівнює їх з відповідними параметрами адміністративної облікового запису (admin/admin). Якщо ім'я користувача та / або пароль не збігаються, виводиться повідомлення про помилку. У разі правильно введених імені користувача та пароля надається доступ до адміністративного поділу.

```php
<html>
  <head>
    <meta charset="utf-8">
    <title>Diplom website</title>
    <meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0" />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js"></script>
  </head>
  <body>
   <?php
  if (isset($_GET['logged'])) {
    $logged = $_GET['logged'];
  } else {
    $logged = 0;
  }
  $error = 0;
  if (isset($_GET['username']) && isset($_GET['password'])) {
    if (strcmp($_GET['username'], 'admin') == 0 && strcmp($_GET['password'], 'admin') == 0) {
      // Successful login
      header('Location: admin.php?logged=1');
      die();
    } else {
      $error = 1;
    }
  }
  if ($logged != 1):
  ?>
    <div class="container py-5">
      <div class="row">
        <div class="col-md-6 mx-auto">
        <?php if ($error == 1): ?>
          <div class="alert alert-danger alert-dismissible fade show" role="alert">
            <strong>ПОМИЛКА!</strong> Неправильний логін або пароль.
            <button type="button" class="close" data-dismiss="alert" aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
        <?php endif; ?>
          <div class="card rounded-0">
            <h3 class="card-header">АВТОРИЗАЦИЯ</h3>
            <div class="card-body">
              <form role="form" action="admin.php" method="get">
                <div class="form-group">
                  <label for="username">Логин</label>
                  <input type="text" class="form-control rounded-0" name="username" id="username" placeholder="Username" required>
                </div>
                <div class="form-group">
                  <label for="password">Пароль</label>
                  <input type="password" class="form-control rounded-0" name="password" id="password" placeholder="Password" required>
                </div>
                <button type="submit" class="btn btn-success rounded-0">Авторизация</button>
              </form></div></div> </div> </div> </div>
    <?php else: // Successful login ?>
    <div class="container py-5"><div class="row"><div class="col-md-6 mx-auto"><div class="card rounded-0">
            <h3 class="card-header">что-то</h3> <div class="card-body"> <p class="card-text">привествую<strong>адмін</strong>!</p>
              <p class="card-text"><a href="admin.php">Log Out</a></p>
     </div></div></div></div></div>
  <?php endif; ?>
  </body>
</html>
```

Рисунок 3.3 – Код додатку авторизації

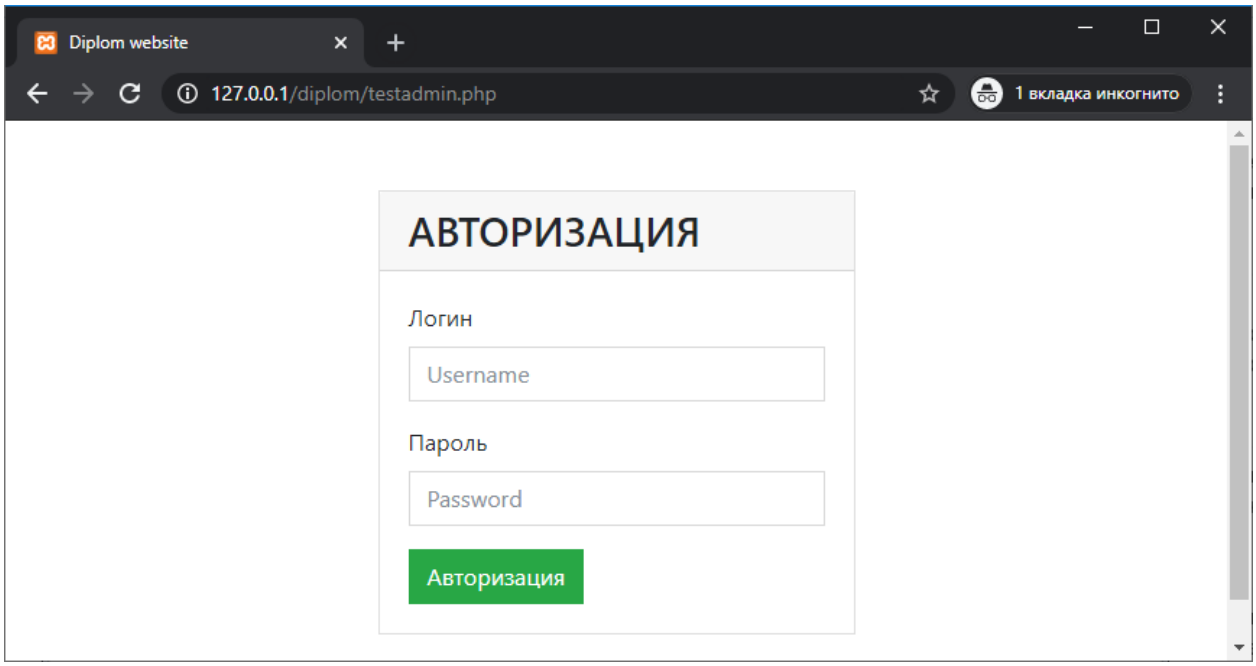

Рисунок 3.5 – Приклад роботи розробленого додатку

3.3 Короткий опис реалізації

Як було описано в другому розділі SIEM система складається з двох частин, колектора, який буде збирати інформацію, та ядра кореляції, який буде приймати рішення.

Розробка колектора для сервера. Потрібно відстежити звернення до розробленої сторінки і спробувати виявити підозрілі дії користувачів. Простий і зрозумілий спосіб - переглянути журнал доступу (журнал звернень) веб-сервера. У цьому випадку разі використовуваного оточення журнал звернень зберігається в файлі access.log.

Загальний опис алгоритму колектора:

- На початковому етапі звертаємося до журналу доступу access.log і запам'ятовуємо розмір файлу.
- Далі в нескінченному циклі з паузою між ітераціями відстежуємо зміни розміру файлу. При збільшенні розміру читаємо з файлу останні додані рядки і передаємо в чергу повідомлень RabbitMQ, запам'ятовуємо новий розмір файлу.
- Файл access.log може бути перезаписаним. Враховуємо такий випадок

(зменшення розміру файлу).

- Запуск програми ApConnector. Колектор починає відстежувати зміни розміру файлу access.log.

Отже, до цього етапу ми налаштували підсистему збору подій безпеки від одного джерела - сервера Apache. Колектора програма ApConnector відстежує зміни журналу звернень access.log і відправляє останні рядки в чергу брокера повідомлень RabbitMQ.

Для опису правил, ядра кореляції використовувано синтаксичні правила корекції, представлені в проектах OSSEC [15], з мінімальними змінами

Моделюється спрощений сценарій атаки типу «перебір пароля» відносно якого захищається додаток, для цього прикладу складемо відповідній набір правил test_rules_for_monitoring.xml (Рис 3.6).

```
<group name="web-app">

  <rule id="100000" level="0">
    <match>/buggy/</match>
    <description>Доступ до додатку</description>
  </rule>

  <rule id="100001" level="0">
    <if_sid>100000</if_sid>
    <match>password</match>
    <description>Спроба авторизації</description>
  </rule>

  <rule id="100002" level="1" frequency="3" timeframe="5">
    <if_matched_sid>100001</if_matched_sid>
    <same_source_ip/>
    <description>Спроба брут-форсу</description>
  </rule>

</group>
```

Рисунок 3.6 – Набір правил для SIEM

Елемент <rule> описує правило. Атрибут id елемента <rule> визначає ідентифікатор правила. Ідентифікатори вибираємо з діапазону, рекомендованого проектом OSSEC для «авторських» правил:> = 100000.

Атрибут level елемента <rule> визначає рівень значимості правила. Мінімальне значення - 0 (в загальному випадку не відображаємо в консолі адміністратора безпеки), максимальне значення - 16. Елемент <match> задає підрядок для пошуку в рядку оброблюваного повідомлення. Елемент <description> задає опис правила, яке буде відображатися в якості оповіщення для адміністратора безпеки.

Випадок спрацьовування такого правила перевіряється просто - якщо підрядок, зазначена в елементі <match>, буде виявлена в рядку оброблюваного повідомлення, ядро кореляції сформує оповіщення безпеки.

Логіка правила 100000 наступна: сповіщати систему безпеки про всі спроби звернення до веб-додатку. Для цього відстежується підрядок «/ dipl /» у всіх зверненнях до веб-сервера. Рівень значущості у правила 100000 нульовий, критичність звернення відсутня, спрацьовування правила буде використовуватися для побудови більш складних ланцюжків правил.

Правило 100000 ніяких залежностей між подіями безпеки не виявляє і «кореляцій» не виявляє. У цьому сенсі коректніше називати такий запис правилом обробки, а не правилом кореляції.

Друге правило. В описі правила 100001 присутній новий елемент <if_sid> з ідентифікатором правила 100000, який накладає додаткову умову спрацьовування - необхідно, щоб до цього спрацювало правило 100000. Логіка правила 100001: якщо рядок, який обробляється має відношення до доступу до веб-додатка (попередньо спрацювало правило 100000) та при цьому в зверненні до веб-сервера виявлено підрядок «password» (можливо свідчить про передачу пароля в форму введення імені користувача і пароля) , то оповістити систему безпеки про спробу отримання адміністративного доступу. Правило 100001 дозволяє виявляти залежності між окремими подіями безпеки і коректно може називатися правилом кореляції.

Третє правило. В описі правила 100002 присутній новий елемент <if_matched_sid> з ідентифікатором правила 100001, який накладає додаткову

умову спрацьовування - необхідно, щоб правило 100001 спрацювало не менше 3 разів (атрибут frequency = «3» елемента <rule>) протягом останніх 5 секунд (timeframe = «5»). Порожній елемент <same_source_ip /> вказує на те, що при підрахунку спрацьовувань правила, заданого елементом <if_matched_sid>, враховуються тільки спрацювання з співпадаючими IP-адресами джерела. Логіка правила 100002: якщо протягом останніх 5 секунд з одного і того ж адреси зафіксовано безліч спроб (3 і більше) отримання доступу до додатку, то сформувати оповіщення з більш високим рівнем значущості level = «1» про спробу підбору пароля доступу - «Спроба брут-форсу».

Набір правил кореляції для даного прикладу сформований, переходимо до безпосередньої реалізації обробника подій. В кінці прикладу виконаємо покрокову налагодження ядра кореляції і перевіримо, як застосовуються правила кореляції до вступників подій безпеки і які оповіщення при цьому формуються.

Програмна реалізація ядра кореляції являє собою найскладнішу і об'ємну частину прикладу, по суті визначаючи всю логіку обробки подій безпеки розробляється SIEM системи. У загальному вигляді логіка роботи ядра кореляції описується наступним чином: ядро кореляції «слухає» чергу повідомлень SIEM_ConnectorQueue. При надходженні чергового повідомлення (події) ядро намагається застосувати до нього заздалегідь завантажені правила обробки подій (правила кореляції). У разі застосування одного з правил до надійшов події безпеки ядро при необхідності формує інцидент безпеки і зберігає його в колекції alerts сховища даних MongoDB. В додаток винесено основні частини коду.

Для простоти роботи з системою було прийнято рішення розробити зручний метод оповіщення, за допомогою telegram-bota, який при виявленні інциденту, буде інформувати в Telgram повідомленням. Було використано мову програмування Python. Код в додатку.

3.3 Тестування розробленої системи

Для перевірки правильності роботи ядра кореляції необхідно переконатися в тому, що в разі виявлення сценаріїв атак (заданих відповідними правилами) ядро формує інциденти безпеки і зберігає їх в колекції alert сховища даних MongoDB. Основне завдання - оповістити адміністратора безпеки про спробу реалізації сценарію атаки типу «перебір пароля».

Тестовий сценарій в загальному вигляді:

1. Зловмисник відкриває сторінку admin.php і кілька разів намагається підібрати ім'я користувача і пароль адміністратора.
2. Після декількох безуспішних спроб зловмисник підбирає облікові дані адміністратора - «admin/admin» - і отримує доступ.
3. Розроблена SIEM система в результаті обробки зареєстрованих подій безпеки із застосуванням заданих правил кореляції виявляє реалізацію сценарію атаки, формує інцидент безпеки і сповіщає адміністратора в Telegram. Далі пропонується виконати тестовий сценарій і розглянути послідовність дій SIEM системи.

Очищаємо сховище даних. Для цього підключаємося до MongoDB за допомогою Robomongo і видаляємо базу даних monitoringSIEM. Запускаємо розроблений ApMonitoringConnector.

Відкриваємо в браузері розроблений додаток за адресою «http://127.0.0.1/test/admin.php». Протягом 10 секунд робимо три спроби підбору імені користувача та пароля адміністратора.

У вікні програми ApConnector перевіряємо факт читання журналу звернень веб-сервера і передачі зареєстрованих подій безпеки в чергу брокера повідомлень. Перевіряємо в панелі адміністрування брокера повідомлень RabbitMQ (http://127.0.0.1:15672/) наявність навантаження: в черзі SIEM_ConnectQueue повинні з'явитися повідомлення.

Далі в процес обробки вступає ядро кореляції. Запускаємо програму MonitoringSIEM.

Перевіряємо формування сповіщень. Для цього підключаємося до MongoDB за допомогою Robomongo і переглядаємо колекцію документів alerts бази даних testSIEM. Колекція повинна бути непорожньою.

Відкриваємо в Telegram розробленого бота, для адміністратора безпеки. Якщо всі компоненти SIEM системи налаштовано правильно, після виконання тестового сценарію у вікні бота з’являється повідомлення, як на рисунку 3.6.

Виявлені не тільки окремі події, а виділена послідовність дій зловмисника і ідентифікований сценарій атаки. Якщо оповіщення з нульовим рівнем значущості розглядати як помилкові спрацьовування, можна відключити їх вивід. І знизити тим самим навантаження на адміністратора.

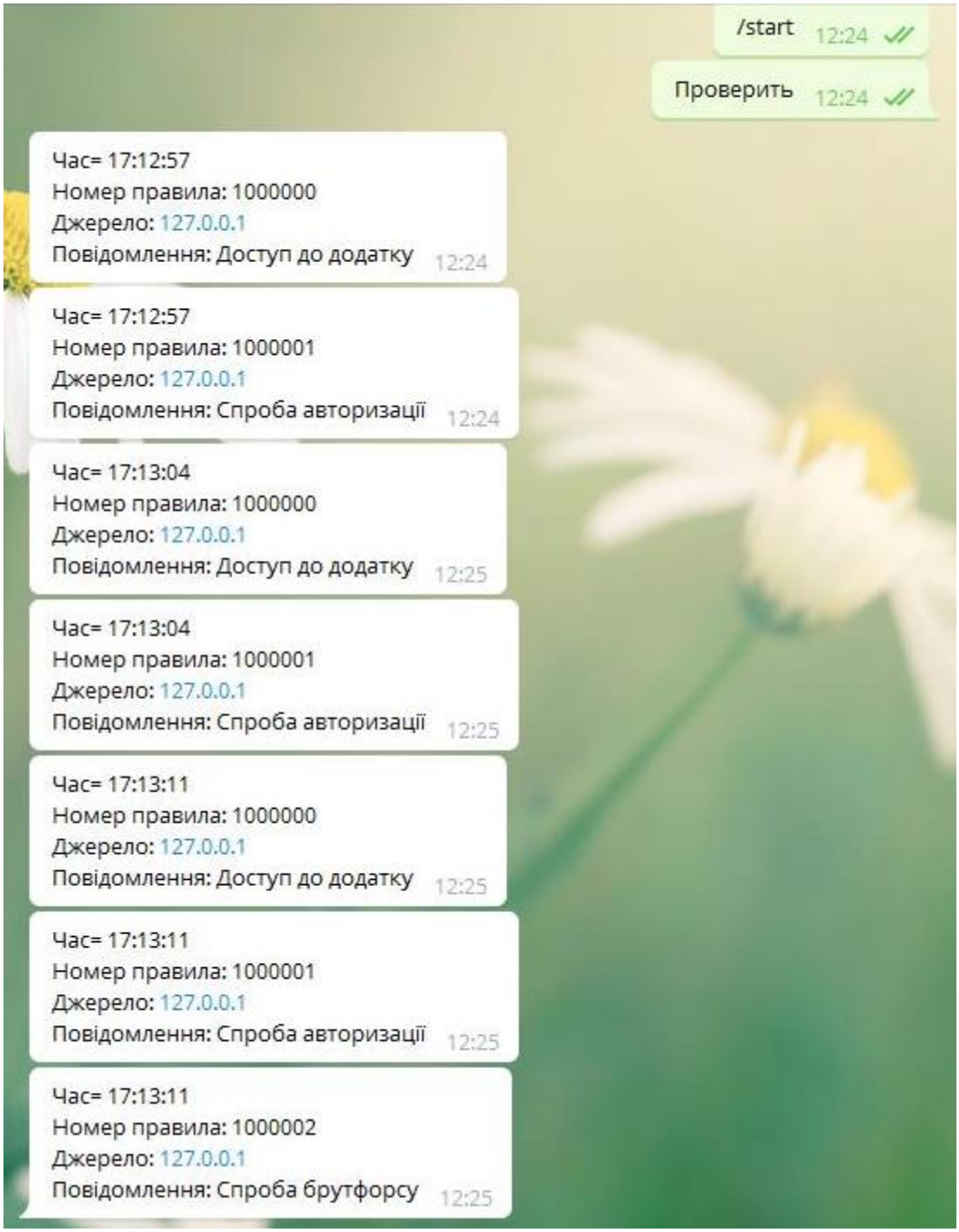

Рисунок 3.6 Приклад роботи

ВИСНОВКИ

Було розглянуто питання побудованих сучасних систем управління інформаційної безпеки (SIEM), для створення комплексного рішення моніторингу безпеки мережі.

У випускній роботі була спроектована і реалізована система SIEM, яка може бути використання для моніторингу безпеки в локальній мережі. Розроблена експертна система може бути використана в мережах будь-якої складності, та на веб-серверах. Продемонстровані можливості застосування розробленої системи в умовах моніторингу безпеки: успішно виявлений сценарій атаки злочинної діяльності типу «перебір паролі», сформований інцидент, та сповіщений адміністратор.

Надалі планується розширити кількість джерел інформації для моніторингу.

СПИСОК ЛІТЕРАТУРИ

1. Степашкин М.В. Моделі та методика анализу захищеності компютерних мереж на основі створення дерева атаки / Степашкин М.В. // дис. канд. техн. наук. – СПб., 2007. – С. 193.
2. Chirillo J. Hack Attack Testing: How to Conduct Your Own Security Audit./ Chirillo J. // – Wiley Publishing, 2003. – P. 576.
3. НД ТЗІ 2.7-009-09 Методичні вказівки з оцінювання функціональних послуг безпеки в засобах захисту інформації від несанкціонованого доступу
4. Семенов Ю.А. Віртеальні локальні мережі VLAN, Интранет [Електронний ресурс] // Телекоммуникационные технологии: сайт. — Режим доступу: http://book.itep.ru/6/vlan_62.html
5. Ковальов Д.О. Виявлення порушень інформаційної безпеки за даними моніторингу інформаційно-телекомунікаційних мереж: дис. канд. техн. наук. - М., 2011. - С. 170.
6. Калинин М.О. Адаптивное управление безопасностью информационных систем на основе логического моделирования : дис. д-ра техн. наук. – СПб., 2010. – С. 310.
7. Хорошко В.А., Методи і інструмети захисту інформації. / Хорошко В.А., Чекатков А.М. // К.: Юниор, 2003. – С. 504.
8. Хайкин С. Нейронные сети: полный курс / Хайкин С. // – М.: Вильямс, 2006. – 2-е изд. – С. 1104.
9. Сторожук Д.О. Методы и алгоритмы для систем мониторинга локальных сетей / Сторожук Д.О. // дис. канд. техн. наук. – М., 2008. – С. 121.
10. Шабуров А.С., Борисов В.И. О применении сигнатурных методов анализа информации в SIEM-системах // Вестник УрФО. Безопасность в информационной среде. – Челябинск: Изд. центр ЮУрГУ, 2015. – № 17. –

С. 23–37. 2. Об информации, информационных технологиях и о защите информации: Федеральный закон от 27 июля 2006 г. № 149-ФЗ

11. SIEM-системы. Инфобезпека [Електронний ресурс] // – Режим доступу: http://www.infobezpeka.com/publications/SIEM_osobennosti_siem
12. Панасенко А. Основные возможности SIEM, исследовательский центр Anti-Malware [Электронный ресурс]. – 2015. Режим доступу: https://www.anti-malware.ru/reviews/
13. Кузнецов А., Федоров А. Современные тенденции развития SIEM-решений, «StorageNews». – № 2(54) [Електронний ресурс] Режим доступу: http://www.ntc-vulkan.ru/images/stories/publication/Vulkan_IS_54- 9_final.pdf
14. K. Kavanagh, T. Bussa, G. Sadowski. Magic Quadrant for Security Information and Event Management / K. Kavanagh, T. Bussa, G. Sadowski // Gartner, 3 December 2018
15. OSSEC documentation. Rules Syntax [Електронний ресурс] // Режим доступу: http://ossec-ocs.readthedocs.io/en/latest/docs/syntax/head_rules.html

ДОДАТОК

ApaConnector.cs

```
using RabbitMQ.Client;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Threading.Tasks;
namespace ApaConnector
{
    Class diplom
    {
        static void Main(string[] args)
    {
        int timeout = 500;
        string connectoName = "ApaConnector";
        string logName = @"c:\xampp\apache\logs\access.log";
        string rabbUri = "amqp://siemuser:siempass@h127.0.0.1/";
        string rabbQueueName = "diplomSIEM_ConnectorQueue";
        WatchLogFile(logName, timeout, connectoName, rabbUri, rabbQueueName);
        Console.ReadLine();// Запобігання завершенню процесу
    }
    static async void seeLogFile(string logName, int timeout,
    string connectoName, string rabbUri, string rabbQueueName)
    {
        Console.WriteLine("Лог-файл: {0}", logName);
        Console.WriteLine("Просмотр лог-файла..." + Environment.NewLine);

        try
        {
            long lFileSize = 0;
            // Запам’ятайте поточний розмір файлу для відстеження змін
            using (FileStream fs = new FileStream(logFileName, FileMode.Open,
                FileAccess.Read, FileShare.ReadWrite))
            {
                lFileSize = fs.Length;
            }
            // Переглянути лог файл

            while (true)
            {
                Console.WriteLine(DateTime.Now.ToString("H:mm:ss"));
                List<string> messList = new List<string>();
                using (FileStream filestream = new FileStream(logName, FileMode.Open,
                    FileAccess.Read, FileShare.ReadWrite))
                {
                    // Set stream position
                    long nFileSize = filestream.Length;
                    if (nFileSize >= lFileSize) filestream.Position = lFileSize;
                    if (nFileSize < lFileSize) lFileSize = 0;
                    // Прочитати останні рядки
                    using (StreamReader sr = new StreamReader(fs))
                    {
                        string nFileSize = null;
                        nFileSize = sr.ReadToEnd();
                        if (nFileSize != null)
                        {
                            lFileSize = nFileSize;
                            if (nFileSize.Length > 0)
                                messList.AddRange(nFileSize.Split(new string[] { Environment.NewLine },
                                    StringSplitOptions.RemoveEmptyEntries));
                        }
                    }
                    // Надіслати нові рядки RabbitMQ
                    if (messList.Count > 0)
                    {
                        ConnectionFactory factor = new ConnectionFactory() { Uri = new Uri(rabbUri) };
                        using (var rConnection = factor.CreateConnection())
                        using (var channel = rConnection.CreateModel())
                        {
                            channel.QueueDeclare(queue: rabbQueueName, durable: false,
                                exclusive: false, autoDelete: false, arguments: null);
                            foreach (var message in messList)
                            {
                                byte[] body = Encoding.UTF8.GetBytes(connectoName + ":" + message);
```

```
                    channel.BasicPublish(exchange: "", routingKey: rabbQueueName,
                        basicProperties: null, body: body);
                }
            }
            // Виведення статистики
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine("  Рядки відправленні  в RabbitMQ: {0}", messList.Count);
            Console.WriteLine(" Останній рядок: {0}", messList[messList.Count - 1]);
            Console.WriteLine(Environment.NewLine);
        }
    }
    // Delay without blocking a thread
    await Task.Delay(timeout);
}
}
catch (Exception ex)
{
    Console.WriteLine("Виключення : {0}", ex.ToString());
}
}
}
```

TelegrambotSIEM.py

```
import pymongo
from pymongo import MongoClient
import telebot
#підключення до бази даних
client = MongoClient()
client = MongoClient('localhost', 27017)
db = client['test-database']
collection = db.test_collection
collection = db['allerts']
cursor = collection.find();
bot = telebot.TeleBot('ваш секретний токен')
@bot.message_handler(commands=['start'])
def start_message(message):
bot.send_message(message.chat.id, 'Телеграм /start')
@bot.message_handler(content_types=['text'])
def send_text(message):
    if message.text =='Проверить' :
    for alert in cursor:
        text="Час="+alert['timestamp']+ "\nНомер правила:"+alert['matching_rule_SID']+ \
             "\nIP відправника"+ alert['src_IP']+"IP отримувача"+ alert['dest_IP']+\
             "Повідомлення:"+ alert['message']
        bot.send_message(message.chat.id, text)
bot.polling()
```

coreSIEM.cs

```
using NLog;
using System;
using System.Collections.Generic;
using System.IO;
using System.Xml;

namespace SIEM
{
    public class CorreEng
    {
        private static Logger log = LogManager.GetCurrentClassLog();

        public int mCount = 10000; // Максимальне значеня в лічильнику matchL
        public Dictionary<int, FireQueue> fDictionar = new Dictionary<int, FireQueue>();
        public Dictionary<int, Rule> ruleList = new Dictionary<int, Rule>();
        public Dictionary<int, int> matchL = new Dictionary<int, int>();

        // Використовується для передачі сповіщень на обробку
        public delegate void ReceivAler(SecurityEvent securityEvent, Rule rule);
        public event ReceivAler onAlerReceiv;

        public void ParseRuleDir(string ruleP)
        {
            int fileNum = 0;
            int ruleN = 0;
            log.Debug("Парсинг правила початок: {0}", ruleP);

            try
            {
                if (!ruleP.Equals(""))
                {
                    string[] fileEntr = Directory.GetFiles(ruleP, "*.xml", SearchOption.AllDirectories);
                    ruleList = new Dictionary<int, Rule>();

                    foreach (string fileName in fileEntries)
                    {
                        FileInfo file = new FileInfo(fileName);
                        List<Rule> fileRuleList = new List<Rule>();
                        ParseRulesFromXML(file.FullName, ref fileRuleList);

                        foreach (Rule rule in fileRuleList)
                            ruleList.Add(rule.ID, rule);

                        fileN++;
                        ruleN += fileRuleList.Count;
                    }
                }
            }
            catch (Exception ex)
            {
                log.Warn("ParseRuleDir exception: {0}", ex.ToString());
            }

            log.Trace("ParseRuleDir: total {0} files processed", fileN);
            log.Trace("ParseRuleDir: total {0} rules processed", ruleN);
            log.Debug("ParseRuleDir stop");

            CheckDependencies(ref ruleList);
        }

        public void ParseRulesFromXML(string fileName, ref List<Rule> ruleList)
        {
            // Синтаксис правил: http://ossec.github.io/docs/syntax/head_rules.html


            string logString = "";
```

```
        try
        {
            log.Trace("ParseRulesFromXML handles file: " + fileName);

            FileInfo file = new FileInfo(fileName);

            // Файл правила може бути недійсним XML, може бути кілька кореневих елементів
            string xml = File.ReadAllText(fileName);
            xml = "<root>" + xml + "</root>";

            // Process variables
            using (XmlReader preReader = XmlReader.Create(new StringReader(xml)))
            {
                while (preReader.ReadToFollowing("var"))
                {
                    string varName = preReader.GetAttribute("name");
                    string varValue = preReader.ReadString();

                    // Replace variable name with its value
                    xml = xml.Replace("$" + varName, varValue);
                }
            }

            XmlDocument doc = new XmlDocument();
            doc.LoadXml(xml);


            // Кожен файл об'єднує правила в групу, дозволено лише атрибут імені
            // src / analyd / rule.c
            XmlNode groupNode = doc.SelectNodes("root/group")[0];
            string groupName = groupNode.Attributes["name"].Value.ToString();

            // Папсинг XML
            foreach (XmlNode node in doc.SelectNodes("root/group/rule"))
            {
                Rule rule = new Rule();
                rule.children = new List<int>();

                rule.ID = (node.Attributes["id"] == null) ? 0 : int.Parse(node.Attributes["id"].Value);
                logString += rule.ID + " ";

                rule.level = (node.Attributes["level"] == null) ? 0 : int.Parse(node.Attributes["level"].Value);
                rule.frequency = (node.Attributes["frequency"] == null) ? 0 : int.Parse(node.Attributes["frequency"].'
                rule.timeFrame = (node.Attributes["timeframe"] == null) ? 0 : int.Parse(node.Attributes["timeframe"].'

                rule.ifSID = (node.SelectSingleNode("if_sid") == null) ? 0 : int.Parse(node.SelectSingleNode("if_sid"
                rule.ifMatchedSID = (node.SelectSingleNode("if_matched_sid") == null) ? 0 : int.Parse(node.SelectSing

                rule.sourceIP = (node.SelectSingleNode("srcip") == null) ? "" : node.SelectSingleNode("srcip").InnerT
                rule.destIP = (node.SelectSingleNode("dstip") == null) ? "" : node.SelectSingleNode("dstip").InnerTex
                rule.sameSourceIP = (node.SelectSingleNode("same_source_ip") == null) ? false : true;

                rule.match = (node.SelectSingleNode("match") == null) ? "" : node.SelectSingleNode("match").InnerText
                rule.description = (node.SelectSingleNode("description") == null) ? "" : node.SelectSingleNode("descr
                rule.parentFile = file.Name;
                rule.XML = node.OuterXml;

                ruleList.Add(rule);
            }

            logString = logString.Trim().Replace(" ", ", ");
            log.Trace("  {0} rules processed: {1}", ruleList.Count, logString);
        }
        catch (Exception ex)
        {
            log.Warn("ParseRulesFromXML exception (log {0}): {1}", logString, ex.ToString());
        }
    }

    public void CheckDependencies(ref Dictionary<int, Rule> ruleList)
```

```
public void CheckDependencies(ref Dictionary<int, Rule> ruleList)
{
    log.Debug("CheckDependencies start");

    // Позначення нащадків у дереві правил
    foreach (KeyValuePair<int, Rule> ruleKVP in ruleList)
    {
        Rule rule = ruleKVP.Value;

        if (rule.ifSID != 0)
            ruleList[rule.ifSID].children.Add(rule.ID);
        if (rule.ifMatchedSID != 0)
            ruleList[rule.ifMatchedSID].children.Add(rule.ID);
    }

    log.Trace("Dependencies: ");

    foreach (KeyValuePair<int, Rule> ruleKVP in ruleList)
    {
        string logString = "";
        foreach (int item in ruleKVP.Value.children) logString += item + " ";
        log.Trace("  {0} children => {1}", ruleKVP.Value.ID, logString.Trim().Replace(" ", ", "));
    }

    log.Debug("CheckDependencies stop");
}

public void GenerateQueueList(Dictionary<int, Rule> ruleList,
    ref Dictionary<int, FireQueue> queueDictionary)
{
    // Створення FireQueues для всіх правил з ненульовою частотою
    log.Debug("GenerateQueueList start");
```

```
    try
    {
        foreach (KeyValuePair<int, Rule> ruleKVP in ruleList)
        {
            // KVP.Value - оброблене правило
            if (ruleKVP.Value.frequency == 0) continue;


            // QueueDictionary.ID = ідентифікатор від елемента ifMatchedSID
            int ID = ruleKVP.Value.ifMatchedSID;

            // Якщо  ID існує
            if (queueDictionary.ContainsKey(ID))
            {
                // тоді обновити timeFrame значення
                queueDictionary[ID].timeFrame = Math.Max(ruleKVP.Value.timeFrame,
                    queueDictionary[ID].timeFrame);
            }
            else
            {
                // else створити новий FireQueue обьект
                FireQueue queue = new FireQueue(ID, ruleKVP.Value.timeFrame);
                queueDictionary.Add(ID, queue);
            }
        }

        log.Trace("Created {0} queues:", queueDictionary.Count);
        foreach (KeyValuePair<int, FireQueue> kvp in queueDictionary)
            log.Trace("  " + kvp.ToString());
    }
    catch (Exception ex)
    {
        log.Warn("GenerateQueueList exception: " + ex.ToString());
```

```
        }

        log.Debug("GenerateQueueList stop");
    }

    public void ProcessMessage(SecurityEvent securityEvent)
    {

    // Перший крок рекурсії
    // Застосовувати лише правила "без батьків"
        foreach (KeyValuePair<int, Rule> ruleKVP in ruleList)
        {
            Rule rule = ruleKVP.Value;
            if (!rule.HasParent()) ApplyRule(securityEvent, rule);
        }
    }

    public void ApplyRule(SecurityEvent securityEvent, Rule rule)
    {
        string padding = Assistant.GetPadding();
        log.Trace("{0}Check rule {1} - {2}", padding, rule.ID, rule.description);

        if (CheckIfMatched(ref securityEvent, ref rule) == true)
        {
            // Виконуються дії, коли правило спрацьовує
            MatchRule(securityEvent, rule);

            //  правила обробляються Норекурсивно
            if (rule.HasChidren())
            {
                log.Trace(padding + "  Check the child rules");
                foreach (int item in rule.children)
                    ApplyRule(securityEvent, ruleList[item]);
                log.Trace(padding + "  Check the child rules: OK");

            }
        }
        else
        {
            log.Trace("{0}Rule {1} not matched", padding, rule.ID);
        }

        log.Trace("{0}Check rule {1}: OK", padding, rule.ID);
    }

    public bool CheckIfMatched(ref SecurityEvent securityEvent, ref Rule rule)
    {
        string padding = Assistant.GetPadding();

        // Перевірка <if_sid> елемента => сканування залежносте й та matchL
        if (rule.ifSID != 0)
        {
            log.Trace("{0}Check <if_sid>{1}</if_sid>", padding, rule.ifSID);

            if (!matchL.ContainsKey(rule.ifSID)) return false;
            if (matchL[rule.ifSID] == 0) return false;
        }

        // Перевірка <same_source_ip> елеиента
        if (!rule.sourceIP.Equals(""))
        {
            log.Trace(padding + "  Check <same_source_ip/>");
            if (!securityEvent.srcIP.Equals(rule.sourceIP)) return false;
        }

        // Перевірка <match> елемента
        if (!rule.match.Equals(""))
        {
            log.Trace("{0}Check <match>{1}</match>", padding, rule.match);
```

```
            bool check = false;
            string[] parts = rule.match.Split(new Char[] { '|' }, StringSplitOptions.RemoveEmptyEntries);

            foreach (string part in parts)
            {
                if (securityEvent.message.IndexOf(part, StringComparison.OrdinalIgnoreCase) >= 0)
                {
                    check = true;
                    break;
                }
            }

            if (!check) return false;
        }

        // Перевырка <if_matched_sid> елемент
        if (rule.ifMatchedSID != 0)
        {
            log.Trace("{0}Check <if_matched_sid>{1}</if_matched_sid>", padding, rule.ifMatchedSID);

            if (matchL.ContainsKey(rule.ifMatchedSID))
            {
                if (matchL[rule.ifMatchedSID] == 0) return false;

                // Сканування FireQueue
                if (!fDictionar.ContainsKey(rule.ifMatchedSID)) return false;
                if (fDictionar[rule.ifMatchedSID].CheckIfMatched(securityEvent, rule))
                {
                    log.Trace("{0}  Rule {1} QueueDictionary.CheckIfMatched == TRUE", padding, rule.ifMatchedSID);
                }
                else
                {
                    log.Trace("{0}  Rule {1} QueueDictionary.CheckIfMatched == FALSE", padding, rule.ifMatchedSID);
                    return false;
                }
            }
        }

        // Співпадіння правила
        return true;
    }
    public void MatchRul(SecurityEvent securityEvent, Rule rule)
    {
        // підняти подію
        onAlertReceived(securityEvent, rule);

        // Якщо правило відслідковується, збільшити лічильник
        if (fDictionar.ContainsKey(rule.ID))
        {
            log.Trace(Assistant.GetPadding() + "  Matched rule is queue tracked");
            fDictionar[rule.ID].Enqueue(securityEvent);
        }
        // зберегти зтатистику
        if (matchL.ContainsKey(rule.ID))
        {
            matchL[rule.ID]++;
            if (matchL[rule.ID] > maxCou) matchL[rule.ID] = maxCou;
        }
        else
            matchL.Add(rule.ID, 1);
    }

    public void LogStatist()
    {
        log.Debug("Статистика співпадінь: ");
        foreach (KeyValuePair<int, int> kvp in matchL)
            log.Trace("  Rule {0} fires {1} times", kvp.Key, kvp.Value);
    }
}
```