Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ВРАЗЛИВОСТЕЙ СЕРВЕРНИХ ОПЕРАЦІЙНИХ СИСТЕМ ЗА ДОПОМОГОЮ ДАНИХ З ВІДКРИТИХ ДЖЕРЕЛ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-3
Білобородов О.Ю.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Любченко В.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Білобородову Олександру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка застосунку для аналізу вразливостей серверних операційних систем за допомогою даних з відкритих джерел

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 26 грудня 2023 р.

3. Вихідні дані до роботи науково-методична література, дані мережі Інтернет, наукові статті, мова програмування C# та платформа .NET, офіційна документація EntityFramework, офіційна документація Docker, офіційна документація Metabase.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд стану питання інформаційної безпеки вебзастосунків.
2. Дослідження наявності даних про вразливості у відкритих джерелах.
3. Програмна реалізація отримання даних про вразливості з відкритих джерел.
4. Розробка методу обчислення безпеки операційних систем.
5. Програмна симуляція роботи системи прийняття рішень для вибору найбільш безпечної операційної системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми вразливостей програмного забезпечення, джерела інформації про вразливості, опис ETL-процесу обробки даних про вразливості, схема бази даних для побудови аналітичних запитів, діаграма класів застосунку, графіки вразливостей, графік симуляції роботи системи автоматизованого прийняття рішень.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	03.11.23-05.11.23	
3	Аналіз літератури з досліджуваної проблеми	05.11.23-07.11.23	
4	Проєктування програмного застосунку	07.11.23-09.11.23	
5	Розробка методу прийняття рішень	09.11.23-11.12.23	
6	Програмна реалізація	11.11.23-17.11.23	
7	Оформлення пояснювальної записки	17.12.23-25.11.23	
8	Перевірка на плагіат	05.12.2023	
9	Рецензування	10.12.2023	
10	Підготовка презентації та доповіді	15.12.2023	
11	Занесення роботи в електронний архів	04.01.2024	
12	Попередній захист кваліфікаційної роботи	04.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Любченко В.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 63 с., 6 табл., 22 рис., 3 дод., 41 джерело.

ІНФОРМАЦІЙНА БЕЗПЕКА, ВРАЗЛИВОСТІ, ЕКСПЛОЙТИ, NVD, CVE, ВІДКРИТІ ДЖЕРЕЛА ПРО ВРАЗЛИВОСТІ, ETL, ОПЕРАЦІЙНІ СИТСТЕМИ.

Об’єктом дослідження є серверні операційні системи та їх вразливості.

Метою дослідження є розробка програмного засобу для статистичного аналізу вразливостей серверних операційних систем та розробка аналітичного методу для прийняття рішень по вибору найбільш безпечної операційної системи.

Нині інформація про вразливості публікується загальнодоступних базах даних і може бути використана для аналізу інформаційної безпеки. В роботі пропонується програмна реалізація для збору даних про вразливості й експлойти з метою подальших аналітичних досліджень, а також проведена програмна симуляція системи, яка в кожний проміжок часу вибирає найбільш безпечні програмні компоненти для своєї роботи.

Перспективою подальших досліджень є масштабування запропонованого підходу для аналізу інших компонентів вебсистеми, тобто серверів баз даних, вебсерверів, серверів вебзастосунків та ін.

INFORMATION SECURITY, VULNERABILITIES, EXPLOITS, NVD, CVE, OPEN SOURCE VULNERABILITIES DATA FEEDS, ETL, OPERATING SYSTEMS.

The object of research is server operating systems and their vulnerabilities.

The purpose of the study is to develop a software tool for statistical analysis of server operating system vulnerabilities and to develop an analytical method for making decisions on choosing the most secure operating system.

Currently, information about vulnerabilities is published in publicly available databases and can be used to analyze information security. The paper proposes a software implementation for collecting data on vulnerabilities and exploits for further analytical research, as well as a software simulation of a system that selects the most secure software components for its operation at each time interval.

The prospect of further research is to scale the proposed approach to analyze other components of the web system, i.e. database servers, web servers, web application servers, etc.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ − програмне забезпечення

ОС − операційна система

NVD – National Vulnerability Database (Національна база даних вразливостей)

CVE – Common Vulnerabilities and Exposures (база даних загальновідомих вразливостей та дефектів безпеки)

CWE – Common Weakness Enumeration (система категорій слабких місць)

CVSS – Common Vulnerability Scoring System (загальна система оцінки вразливостей)

CPE – Common Platform Enumeration (структурована схема іменування систем, програмного забезпечення та пакетів інформаційних технологій)

Data Lake – озера даних

DWH – Data Warehouse (сховище даних)

ETL – Extract, Transform, Load (Витягування, перетворення та завантаження)

NIST – National Institute of Standards and Technology (Національний інститут стандартів і технологій)

GSA – U.S. General Services Administration (Адміністрація загальних служб США)

CISA – Cybersecurity & Infrastructure Security Agency (Агентство з кібербезпеки та безпеки інфраструктури)

ВСТУП

Проблеми інформаційної безпеки є важливими та актуальними для сучасного суспільства. Повальна інформатизація, технологізація охоплює все більше сфер суспільства, від використання розумних помічників у побуті до інформатизації цілих галузей держави. В цьому є вагомі переваги: в побуті – це економія часу на повсякденних рутинних справах, для бізнесу – це можливість економії на ресурсах (наприклад, реалізація концепції «банк без відділень»), для держави перенесення послуг в онлайн – це спрощення взаємодії формальних процедур та ефективний механізм боротьби з корупцією. Роблячи таку цифрову трансформацію, ми вирішуємо одні проблеми, але тепер стикаємося і з новими [1]. Перехід сервісу, бізнесу, або цілої галузі в онлайн означає появлення нових викликів, пов'язаних з необхідністю підтримки таких характеристик інформаційної безпеки [2] як

– доступність (availability);
– цілісність (integrity);
– конфіденційність (confidentiality).

Тобто, говорячи простими словами, інформаційна система повинна справлятися з загрозами, що можуть викликати непрацездатність системи, що призводять до несанкціонованої зміни даних та з загрозами, що викликані розкриттям конфіденційної інформації особам, що не мають на це спеціальних повноважень.

Однією зі складових гарантоспроможності інформаційних систем є безпека [3]. Основну загрозу безпеці вебзастосунків становлять уразливості, що містяться в програмних компонентах.

На сьогоднішній день питанням виявлення та аналізу вразливостей програмного забезпечення приділяється значна увага з боку великих компаній і дослідницьких центрів. Як випливає зі щорічного спільного звіту американського інституту комп'ютерної безпеки та ФБР за 2010 рік [4], 67% опитаних представників великих корпорацій і державних організацій

повідомили про незаконне вторгнення у свої комп'ютерні системи, 17% стали жертвами DoS-атак, 7% організацій повідомили про інформаційні вторгнення в корпоративні вебсистеми, 10% опитаних повідомили про атаки на їхні веббраузери, 13% – про несанкціонований доступ або ескалацію привілеїв третіми особами. Середній збиток, завданий проблемами безпеки, було оцінено на рівні 20 мільйонів доларів.

У 2021 році зросла кількість зловмисників, які використовують уразливості в загальнодоступних програмах, щоб отримати доступ до мереж організацій [5]. Хоча в деяких випадках зловмисники зосереджуються на помилках нульового дня, частіше вони дивляться на нещодавно виправлені вразливості та шукають невиправлені системи.

Яскравим прикладом цього були критичні вразливості в Microsoft Exchange Server, відомі під загальною назвою ProxyLogon. Недоліки були виправлені на початку березня 2021 року, тоді як Microsoft заявила, що помилки використовувалися групою передових постійних загроз (APT), яку вона назвала Hafnium (Symantec відстежує цю групу як Ant) у цілеспрямованих атаках. Однак незабаром після того, як уразливості ProxyLogon були розкриті, інші учасники загроз почали використовувати їх [5].

Це швидке впровадження також було підкреслено, коли в серпні 2021 року було публічно оприлюднено іншу низку вразливостей у сервері Microsoft Exchange Server, яка отримала назву ProxyShell. Спроби експлойту, націленого на ці помилки, почалися негайно, а дані Symantec показують понад 200 000 спроб експлойту, націленого на цей набір уразливостей у серпні. 2021 один.

Інші вразливості в загальнодоступних програмах, якими часто користувалися зловмисники у 2021 році, включають недоліки в продуктах VPN від Pulse Secure (CVE-2019-11510), Fortinet (CVE-2018-13379) і SonicWall (CVE-2021-20016), і вразливості в програмному забезпеченні Accellion File Transfer Appliance (FTA) (CVE-2021-27101, CVE-2021-27102, CVE-2021-27103 і CVE-2021-27104).

1 СУЧАСНИЙ СТАН ПИТАННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ВЕБ ЗАСТОСУНКІВ

1.1 Інформаційна безпека

Інформаційна безпека – це стан захищеності систем обробки і зберігання даних, при якому забезпечено конфіденційність, доступність і цілісність інформації, використання й розвиток в інтересах громадян або комплекс заходів, спрямованих на забезпечення захищеності інформації особи, суспільства і держави від несанкціонованого доступу, використання, оприлюднення, руйнування, внесення змін, ознайомлення, перевірки запису чи знищення (у цьому значенні частіше використовують термін «захист інформації») [6].

Інформаційна безпека вебзастосунків включає в себе наступні аспекти:

– захист конфіденційності даних. Сервери часто містять важливі та конфіденційні дані, такі як особиста інформація користувачів, фінансова інформація, комерційні та наукові дані. Забезпечення безпеки серверних операційних систем гарантує, що ці дані залишаються недоступними для несанкціонованих осіб;

– забезпечення доступності сервісів. Безпека серверних операційних систем також забезпечує доступність сервісів для користувачів. Зловмисники можуть спробувати зробити вебсайти або онлайн-сервіси недоступними для користувачів шляхом атак на сервери. Заходи безпеки допомагають запобігти таким атакам та зберегти доступність сервісів;

– захист від втрати даних. Важливо забезпечити резервне копіювання даних та відновлення систем у разі аварій, природних катастроф або інших непередбачуваних подій. Інформаційна безпека допомагає запобігти втратам даних та забезпечити їхнє відновлення;

– захист від кібератак. Зловмисники можуть використовувати різні методи, такі як віруси, троянці, атаки на мережу, фішинг тощо, для атаки

серверних операційних систем. Інформаційна безпека допомагає виявляти та блокувати такі атаки;

– дотримання нормативних вимог. В багатьох галузях, таких як фінанси, охорона здоров'я, громадська безпека, інформаційна безпека є обов'язковою згідно з законодавством та регулятивними вимогами. Недотримання цих вимог може призвести до серйозних правових наслідків;

– захист від репутаційних ризиків. Порушення безпеки серверних операційних систем може призвести до втрати репутації організації. Користувачі та клієнти вірять у те, що їхні дані та інформація зберігаються в безпечності, і будь-які порушення цієї довіри можуть вплинути на довіру до організації;

– збереження інтелектуальної власності. Велика частина інтелектуальної власності організацій, така як програмне забезпечення, дизайн, патенти та інше, може бути збережена на серверах. Забезпечення безпеки серверних операційних систем допомагає захистити цю інтелектуальну власність від крадіжки та незаконного доступу.

Отже, інформаційна безпека серверних операційних систем відіграє важливу роль у захисті даних, забезпеченні доступності сервісів та запобіганні кібератакам, що допомагає зберегти цілісність та довіру до інформаційних систем та організацій в цілому.

Актуальність дослідження полягає у дослідженні безпеки серверних операційних систем.

1.2 Побудова сучасних клієнт-серверних рішень

Розглядаючи побудову сучасних клієнт-серверних застосунків та SaaS рішень, можна виділити типовий стек для таких рішень, який включає [7]

– вебзастосунок;

– вебсервер та сервер застосунків;

– сховище даних (сервер баз даних);

– операційну систему.

Кожен з цих компонентів може містити вразливості. При цьому, слід зауважити, що якщо розробник може впливати на безпеку вебзастосунків, так як йому доступні усі вихідні коди застосунку, то операційна система, сховище баз даних, вебсервер та сервер застосунків він використовує як готові рішення, що прийнято називати COTS – Component of the shelf, тобто такий, що розробляється сторонніми розробниками та надається «як є». Звісно, розробники операційних систем також піклуються про безпеку свого продукту та приділяють неабияку увагу цьому, але сучасні системи надто складні, що призводить до того, що виявити абсолютно усі вразливості в програмному забезпечені до випуску релізу є неможливою задачею. Розробляють систему люди, та люди можуть помилятися та додавати дефекти в програмне забезпечення, що можуть впливати на інформаційну безпеку. Саме тому питання вивчення вразливостей є дуже важливим на сьогоднішній день.

1.3 Способи атаки на операційні системи

Існує різні методи атак на серверні операційні системи, з основних можна відмітити:

– атаки на використання вразливостей (Vulnerabilities). Ці атаки використовують вразливості операційної системи або програмного забезпечення, які можуть дозволити зловмисникам виконати код на сервері або отримати несанкціонований доступ. Це може включати атаки на вразливості у вебсерверах, протоколах, програмах та службах;

– атаки на отримання аутентифікаційних даних (Authentication Attacks). Спроби отримати логіни та паролі для входу в серверну операційну систему. Це може включати брутфорс атаки, атаки на перехоплення сесій, фішинг і багато інших методів;

– атаки на відмову в обслуговуванні (DoS або DDoS атаки). Атаки на відмову в обслуговуванні спрямовані на перевантаження серверів, щоб вони були недоступні для звичайних користувачів, яка може реалізовуватися як атакою на вразливість, так і відправкою великої кількості запитів одночасно. В такому разі DoS атака використовує один сервер для атаки, у випадку DDoS використовується група атакуючих серверів;

– атаки на перехоплення інформації (Information Interception). Ці атаки спрямовані на перехоплення чутливої інформації під час її передачі між сервером і клієнтом. Це може бути атака «прослуховувача» (sniffing) або атака «людина в середині» (man-in-the-middle), де зловмисник веде спостереження або навіть змінює передачу даних;

– атаки на ідентифікацію і авторизацію (Identity and Authorization Attacks). Ці атаки спрямовані на порушення прав доступу користувачів. Це може включати атаки на перехоплення сесій, обман користувачів або атаки на перехоплення даних авторизації;

– атаки на вміст та даних (Data and Content Attacks). Ці атаки спрямовані на несанкціонований доступ, модифікацію або видалення даних на сервері. Це може включати атаки на вразливості в програмному забезпеченні, яке обробляє дані, а також атаки на бази даних та файли;

– атаки на цілісність даних (Integrity). Ці атаки спрямовані на те, щоб внести несанкціоновані зміни в систему або файли, що використовує ця система.

Ці атаки можуть бути спрямовані на різні рівні операційних систем, включаючи рівень мережі, рівень програмного забезпечення та рівень апаратного забезпечення. Захист від цих атак включає в себе встановлення заходів безпеки на всіх рівнях та ретельне слідкування за забезпеченням інформаційної безпеки.

1.4 Життєвий цикл вразливості програмного забезпечення

Життєвий цикл вразливості програмного забезпечення включає в себе кілька етапів від виявлення вразливості до її виправлення або видалення. Основні етапи життєвого циклу вразливості:

– виявлення вразливості (Discovery). Цей етап полягає в виявленні можливих вразливостей у програмному забезпеченні. Вразливості можуть бути виявлені шляхом аналізу вихідного коду, впровадження тестів на проникнення, моніторингу подій та інших методів. Дослідники безпеки, етичні хакери, аудитори безпеки та інші можуть виявляти вразливості;

– класифікація та оцінка (Classification and Assessment). Після виявлення вразливості вона класифікується, оцінюється за ступенем серйозності та визначається потенційна небезпека, яку вона може створити для системи або даних;

– повідомлення вендору (Vendor Notification). Якщо вразливість була виявлена дослідником, вендор (розробник програмного забезпечення) повинен бути повідомлений про цю вразливість. Зазвичай повідомлення включає в себе технічний опис вразливості та можливий варіант виправлення;

– валідація (Validation). Після отримання повідомлення вендор перевіряє вразливість та визначає, чи вона справді існує та як її виправити;

– розробка патча (Patch Development). Якщо вразливість підтверджена, розробляється патч або оновлення, яке виправлює вразливість. Патч повинен бути протестованим та перевіреним, щоб забезпечити, що він не вносить нові проблеми;

– розголошення (Disclosure). Після створення патча та виправлення вразливості інформація про вразливість може бути опублікована для сповіщення громадськості та інших сторін, які використовують програмне забезпечення. До цього моменту вендор має можливість виправити вразливість;

– виправлення (Remediation). Користувачі та адміністратори систем повинні встановити патч або оновлення для усунення вразливості.

1.5 Методи виявлення вразливостей

Серед популярних методів виявлення вразливостей можна виділити:

– сканування на вразливості (Vulnerability Scanning). Цей метод використовується для автоматичного сканування програмного забезпечення і інфраструктури на предмет виявлення відомих вразливостей. Він може використовувати бази даних відомих вразливостей та сигнатур для пошуку потенційних проблем;

– тестування на проникнення (Penetration Testing). Це активний підхід, коли експерти з кібербезпеки намагаються використати вразливості для здійснення контрольованих атак на систему. Це може включати в себе атаки на введення, спроби взяти під контроль сервери та інші техніки;

– аудит вихідного коду (Code Auditing). Цей метод включає в себе аналіз вихідного коду програмного забезпечення для виявлення потенційних вразливостей. Це може бути вручну виконуваним аудитом або застосуванням автоматизованих інструментів;

– спостереження за трафіком (Traffic Monitoring). Спостереження за мережевим трафіком дозволяє виявляти аномальні активності, які можуть бути пов’язані з атаками або вразливостями в мережевій інфраструктурі;

– аналіз журналів безпеки (Security Log Analysis). Аналіз журналів безпеки інфраструктури та програмного забезпечення може допомогти виявити аномальні дії та події, які можуть вказувати на вразливості або атаки;

– верифікація та валідація даних (Data Validation). Перевірка та валідація даних введених користувачем може запобігти багатьом видам атак, таким як SQL-ін’єкція та XSS;

– аналіз системних налаштувань (Configuration Analysis). Перевірка налаштувань серверів і програм для виявлення помилок налаштування, які можуть створювати вразливості;

– виявлення нових вразливостей (Zero-Days Vulnerabilities). Використання спеціальних інструментів та методів для виявлення раніше невідомих вразливостей, відомих як «вразливості нульового дня»;

– соціальна інженерія (Social Engineering). Аналіз та тестування атак, які використовують маніпуляцію людьми, а не технічні вразливості.

– аналіз внутрішніх загроз (Insider Threat Analysis). Виявлення загроз від власних співробітників або інших внутрішніх джерел.

Ці методи можуть використовуватися окремо або в поєднанні для виявлення вразливостей в програмному забезпеченні і інфраструктурі.

1.6 Аналіз робіт з інформаційної безпеки операційних систем

У роботі [8] розглядається аналіз процесного підходу проведення аудиту інформаційної безпеки Linux-подібних серверних операційних систем (ОС), оскільки саме сервер є найважливішим компонентом корпоративної мережі. Проведення аудиту дозволяє виявити прогалини в системі захисту сервера, які можуть призвести до втрати конфіденційної інформації чи завдати іншої шкоди. На основі отриманих даних, розроблено комплекс рекомендацій направлених на усунення виявлених прогалин і на надійне забезпечення захисту інформації.

У роботі [9] був проведений експеримент та змодельована робота web-сервіса на вибраних програмних компонентах, атаки на цей сервіс і визначено реакцію сервіса на ці атаки. Була вивчена типова архітектура web-сервера, описаний алгоритм роботи web-сервіса, побудований скінчений автомат Мура для отриманого алгоритму. Створена програмна реалізація імітаційної моделі та інтерфейс користувача програми. Визначена імітаційна модель дозволяє

підвищити точність оцінки та дослідити залежність показників гарантоздатності web-сервісів від інтенсивності і кратності атак. Отримувані результати роботи можна застосовувати для вибору такої конфігурації компонентів вебсервісу, яка буде забезпечувати кращу гарантоздатність.

В статті [10] розглянуто вразливості та атаки на державні інформаційні ресурси, що обробляються засобами інформаційно-телекомунікаційних систем для визначення множини параметрів при оцінці захищеності державних інформаційних ресурсів. Розглянуто порушення у сфері використання електронно-обчислювальних засобів, телекомунікаційних систем і комп'ютерних мереж. Представлено загальну структуру реалізації атаки. Проведено аналіз вразливостей інформаційно-телекомунікаційних систем обробки державних інформаційних ресурсів та атак на системи обробки державних інформаційних ресурсів. Розглянуто сучасні бази даних, які містять детальний опис вразливостей та атак. Представлено класифікацію атак та параметри цих атак. Описано стратегії здійснення атак. Розглянуто основні фази та особливості проведення атак. Висунуто перелік вимог до методів виявлення атак. Визначено, що реалізація загроз відбувається за допомогою множини різнонаправлених атак. Запропоновано для визначення множини параметрів при оцінці захищеності державних інформаційних ресурсів, що обробляються засобами інформаційно-телекомунікаційних систем, забезпечити функціонування систем виявлення атак та визначення вразливостей з урахуванням вимог до методів виявлення атак, параметрів даних та характеристичних особливостей сучасних систем виявлення атак.

У статті [11] описується заснована на даних систему оцінки загрози вразливостей, тобто ймовірності того, що вразливість буде використана протягом перших 12 місяців після публічного розкриття. Пропонуються моделі для передбачення появи нових експлойтів на вразливості.

1.7 Постановка задачі дослідження

Таким чином, аналіз вразливостей серверних операційних систем є актуальним завданням для підвищення інформаційної безпеки. Тому ставиться завдання розробки програмного засобу для аналізу вразливостей серверних операційних систем на основі даних про вразливості з відкритих джерел.

Об’єктом дослідження є серверні операційні системи та їх вразливості.

Метою дослідження є розробка програмного засобу для статистичного аналізу вразливостей серверних операційних систем та дослідження можливості побудови системи з найбільш безпечних компонентів.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих джерел про вразливості;

– зібрати інформацію про операційні системи, що досліджуються;

– розробити програмну реалізацію для автоматичної обробки даних з відкритих джерел;

– провести роботи системи прийняття рішень для вибору найбільш безпечної операційної системи.

2 РОЗРОБКА МЕТОДУ СТАТИСТИЧНОГО АНАЛІЗУ ВРАЗЛИВОСТЕЙ НА ОСНОВІ ДАНИХ З ВІДКРИТИХ ДЖЕРЕЛ

2.1 Збір інформації про програмне забезпечення

Перший етап аналізу вразливостей програмного забезпечення полягає в зборі початкової інформації.

На даному етапі необхідно проаналізувати програмні продукти, що досліджується та для кожного відповісти на наступні питання:

– яку версію продукту збираємось аналізувати?

– яка архітектура продукту? Це монолітне ядро чи продукт має модульну архітектуру? Якщо продукт складається з додаткових модулів, то ці модулі також потребують аналізу на вразливості, то ж їх вразливості також мають вплив на безпеку кінцевого продукту;

– період, за який збираємось аналізувати. Кожний програмний продукт проходить певні стадії від релізу до стадії закінчення офіційної підтримки. Якщо мова йде про операційні системи, то необхідно знайти дату виходу в реліз певної версії операційної системи, що досліджується.

Відповіді на ці питання наведемо в таблиці 2.1. Інформація зібрана з відкритих джерел [12-16].

Для дослідження оберемо чотири серверних операційних системи, що вийшли приблизно в той самий проміжок часу. Щоб в нас було достатньо даних для аналізу вразливостей та експлойтів, візьмемо за основу 2008 рік. Таким чином, під наш аналіз попадуть наступні операційні системи:

– Linux Enterprise Server 11;

– Mac OS X Server 5;

– Microsoft Windows Server 2008;

– Novel Linux Server 5.

Таблиця 2.1 – Серверні операційні системи, що досліджуються

Назва	**Дата релізу**	**Архітектура**
Linux Enterprise Server 11	24.03.2009	Монолітне ядро з модулями. Включає ядра – Linux Kernel 2.6.27 – Linux Kernel 2.6.32 – Linux Kernel 3.0.40-3.0.68
Mac OS X Server 5	26.10.2007	Гібридне ядро або макро ядро
Microsoft Windows Server 2008	27.03.2008	Гібридне ядро або макро ядро
Novel Linux Server 5	15.03.2007	Монолітне ядро з модулями. Включає ядра: – Linux Kernel 2.6.18 – Linux Kernel 5.2

2.2 Аналіз джерел інформації про вразливості та експлойти

Нині існує значна кількість урядових, міжнародних і комерційних організацій, що займаються збором інформації про вразливості програмних засобів. Інформація про ці вразливості зберігається в спеціальних, як правило, загальнодоступних базах даних і може бути використана для оцінки рівня безпеки вебсистем [17]. Приклади та коротка характеристика деяких баз даних вразливостей та експлойтів наведені в таблиці 2.2.

Таблиця 2.2 – Порівняння джерел про вразливості та експлойти

№	Інформаційний ресурс	Коротка характеристика	Спосіб надання інформації
1	2	3	4
1	National Vulnerability Database (NVD) http://nvd.nist.gov/	Національна база вразливостей, один із найпередовіших ресурсів вразливостей, який інтегрує всі бази вразливостей, доступні в США, та надається для безкоштовного доступу. NVD базується на інформації CVE і синхронізує всі свої поповнення зі словником CVE. Рейтинг серйозності вразливостей виставляється згідно зі стандартом CVSS.	Вебінтерфейс, XML- дампи, JSON- дампи, API
2	Common Vulnerabilities and Exposures (CVE) http://cve.mitre.org/	Підтримується корпорацією Mitre. Надається в режимі вільного доступу і фінансується урядом США. Головна мета CVE - стандартизувати імена всіх відкритих і публічно відомих вразливостей у програмних продуктах. Основне завдання - полегшити обмін даними між окремими базами вразливостей та інструментами із забезпечення інформаційної безпеки. Більшість наявних баз і ресурсів вразливостей програмних продуктів використовують CVE індекси і синхронізуються з інформацією, доступною в CVE.	Вебінтерфейс, XML-дампи

Продовження таблиці 2.2

1	2	3	4
3	Open Vulnerability and Assessment Language (OVAL). http://oval.mitre.org/	Є міжнародним громадським стандартом інформаційної безпеки. OVAL містить мову, що використовується для кодування подробиць системи та перичень вмісту сховища, прийняту спільнотою. Мова стандартизує три головні оцінки процесу: подання конфігурації системи, що тестується, аналіз системи на наявність певних станів (вразливість, конфігурація, стан "патча") і створення звіту про результати оцінки. Сховище містить загальнодоступну відкриту інформацію про використовувану мову.	Вебінтерфейс
4	Computer Emergency Response Team (CERT) http://www.cert.org/	Використовувані назви: Команда комп'ютерної безпеки з реагування на інциденти (Computer Security Incident Response Team, CSIRT), або Комп'ютерна команда екстреної готовності (Computer Emergency Readiness Team) - назви експертних груп, що займаються інцидентами в комп'ютерній та інтернет-безпеці. У США групи CERT найчастіше співпрацюють з оригінальним CERT з університету Карнегі-Меллон, який служить як загальнонаціональний координаційний центр.	Вебінтерфейс

Продовження таблиці 2.2

1	2	3	4
5	Secunia http://secunia.com/	Данська комерційна організація, яка надає сервіси із забезпечення інформаційної безпеки та пропонує програмні продукти для пошуку комп'ютерних вірусів і вразливостей програмних продуктів. Secunia утримує штат фахівців з безпеки, які тестують, перевіряють, контролюють та оцінюють публічні повідомлення про дефекти та вразливості. Водночас фахівці Secunia проводять і власну роботу з пошуку дефектів і прогалин у безпеці.	Вебінтерфейс
6	Exploit DB https://www.exploit-db.com/	Відкрита база даних експлойтов. Містить короткий опис експлойту, вихідний код та додаткові посилання, як наприклад ідентифікатор вразливості згідно списку бази даних CVE.	Вебінтерфейс, CSV-файли

Порівняння даних, що містяться в базах даних вразливостей, розглянуто в [18].

Найбільш інформативними базами в нашому випадку є NVD та ExploitDB, що надають вміст своїх баз у вигляді файлів для подальшої автоматичної обробки.

Вміст бази даних NVD представляється у вигляді файлів форматів XML та JSON, що розбиті по роках, які можна загрузити на сторінці [19]. Виділимо інформацію, що нас буде цікавити з аналітичної точки зору:

– унікальний ідентифікатор вразливості CVE;

– базова оцінка критичності вразливості;

– короткий опис вразливості (Summary);

– дата публікації та внесення змін про вразливості.

Вміст бази даних Exploit DB доступний у вигляді файлу формату CSV, які можна загрузити на сторінці [20]. Виділимо інформацію, що нас буде цікавити з аналітичної точки зору:

– унікальний ідентифікатор вразливості в базі Exploit DB;

– список ідентифікаторів вразливості CVE;

– короткий опис експлойту;

– дата додавання запису про експлойт, публікації запису та внесення останніх змін до запису.

2.3 Розробка програмної реалізації для обробки статистичної інформації про вразливості

2.3.1 Проектування архітектури програмного застосунку

Задача аналізу даних про вразливості з відкритих джерел є прикладом аналітичної задачі, що передбачає наступні етапи:

– витягевання інформації з відкритих джерел;

– трансформування інформації у структуру даних, зручну для аналізу;

– завантаження даних та аналіз.

Якщо повернутися до аналізу джерел про вразливості, то можна помітити, що джерела мають різний формат вихідних даних, який не зручний для побудови аналітичних запитів. Таким чином, нам необхідно застосувати прийом, що зветься ETL [21] та побудувати власне сховище даних [22] для подальшого аналітичного аналізу.

Рисунок 2.1 відображає основні компоненти системи для аналізу даних з відкритих джерел.

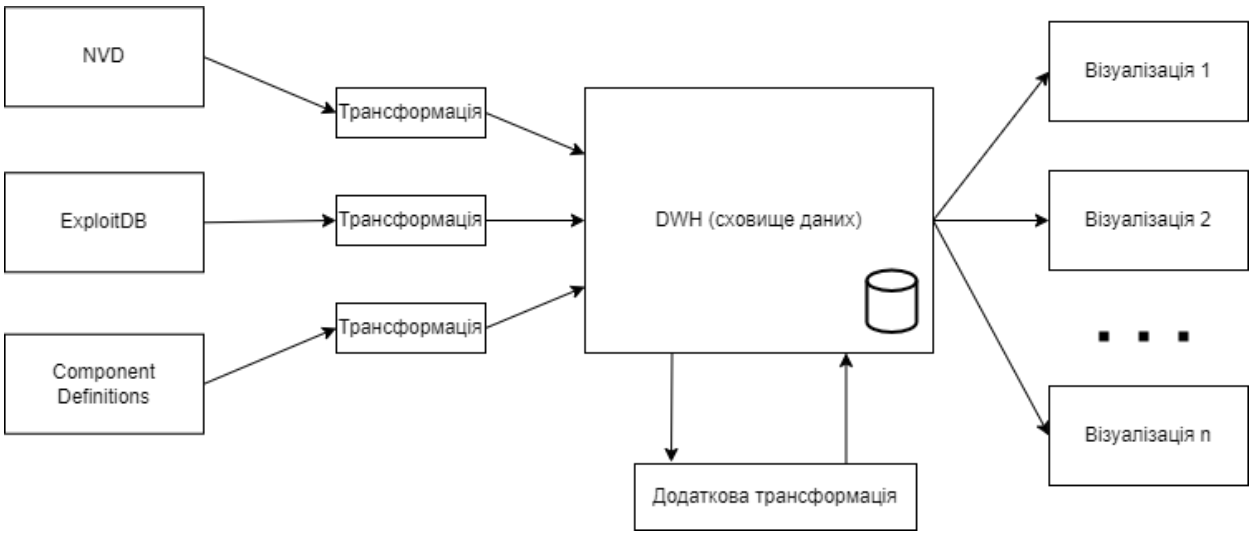

Рисунок 2.1 – Архітектура типової системи для аналізу даних про вразливості

Архітектура типової системи включає в себе

– NVD – зліпок бази даних NVD, надаються у форматі JSON по роках;

– ExploitDB – дампи бази даних ExploitDB, надаються у форматі CSV;

– Трансформація – програмна реалізація, що зчитує вхідні файли у різних форматах та поміщує зчитану інформацію у базу даних сховища;

– DWH (сховище даних) – база даних, що зберігає інформацію в таблицях для подальшої обробки в аналітичних запитах;

– Трансформація з симуляцією – додаткова обробка даних, що зберігаються в сховищі даних для ретроспективного аналізу на основі історичних даних;

– Візуалізація 1, 2 ... n – представлення даних, що зберігаються у сховищі у вигляді графіків, таблиць тощо.

2.3.2 Розробка сховища даних

Відповідно до джерел даних, база даних сховища повинна містити наступні таблиці:

– Vulnerabilities – оброблена інформація про вразливості;

– Exploits – інформація про експлойти;

– CpeIdentifiers – список ідентифікаторів вразливих програмних компонентів;

– ComponentDefinitions – інформація про компоненти, що досліджуються;

Для аналітичних запитів до багатовимірної бази даних застосуємо логічну структуру таблиць під назвою «зірка» [23], що є окремим випадком структури «сніжинка» [24]. Таким чином, в нас з'являються додаткові таблиці фактів:

– Fact_ComponentVulnerability – таблиця фактів про вразливості компонентів, що досліджуються;

– Fact_DecisionSimulations – таблиця фактів з прийнятими рішеннями під час симуляції. Логіка прийняття рішень розглянута в розділі 2.5;

та таблиця вимірів:

– Dim_Date – таблиця вимірів дат.

Рисунок 2.2 демонструє модель даних та зв'язки між таблицями у вигляді UML діаграми відносин сутностей.

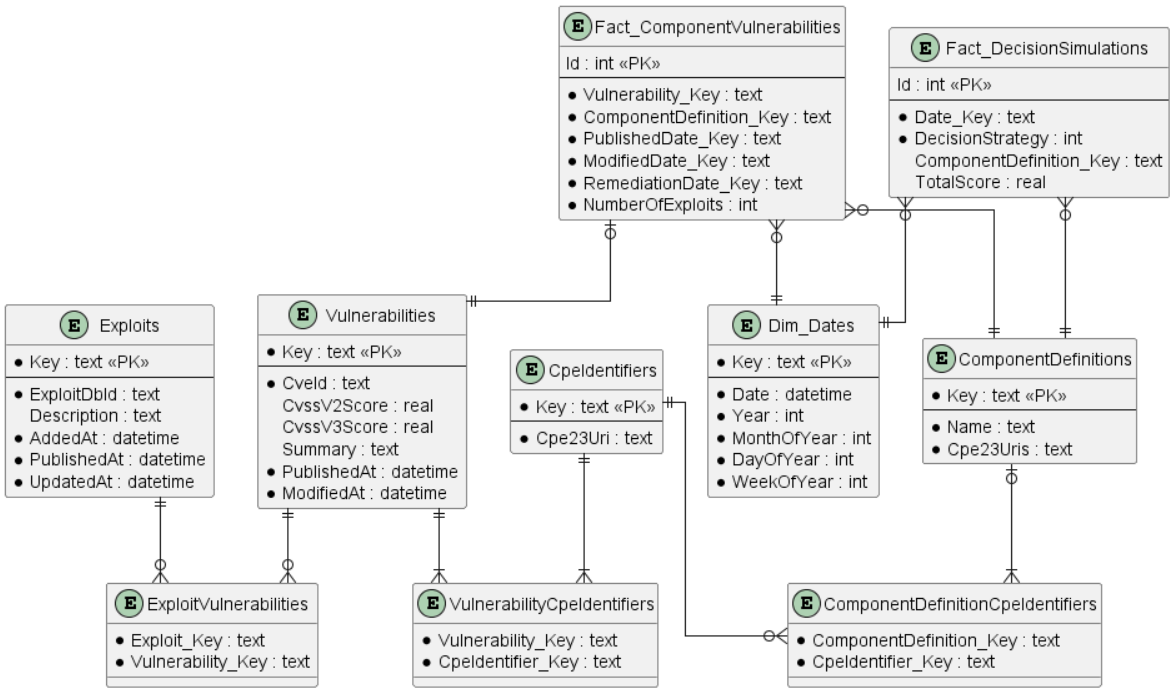

Рисунок 2.2 – Схема даних для аналітичних обчислень

Для оптимізації вставки взаємозалежних сутностей, будемо замість автоматично генерованого послідовного первинного ключа використовувати хеш-функцію Sha128 від ідентифікуючого поля. Опис полів наведено в таблиці 2.3.

Таблиця 2.3 – Опис полів таблиць сховища даних

Поле	**Призначення**
1	**2**
Таблиця Vulnerabilities	
Key	Первинний ключ, отриманий за допомогою функції Sha128(CveId).
CveId	Унікальний ідентифікатор вразливості згідно CVE.
CvssV2Score	Оцінка критичності вразливості згідно специфікації загальної системи оцінки вразливості, викладеної у версії 2 [25].
CvssV3Score	Оцінка критичності вразливості згідно специфікації загальної системи оцінки вразливості, викладеної у версії 3.1 [26].
Summary	Опис вразливості.
PublishedAt	Дата публікації інформації про вразливість в базі даних вразливостей NVD.
ModifiedAt	Дата внесення змін до інформації про вразливість в базі даних вразливостей NVD.
Таблиця Exploits	
Key	Первинний ключ, отриманий за допомогою функції Sha128(ExploitDbId).
ExploitDbId	Ідентифікатор запису у базі даних ExploitDB.
Description	Опис експлойту.

Продовження таблиці 2.3

1	2
AddedAt	Дата додавання запису в базу даних експлойтів.
PublishedAt	Дата публікації запису.
UpdatedAt	Дата внесення останніх змін у запис.
Таблиця ComponentDefinitions	
Key	Первинний ключ, отриманий за допомогою функції Sha128(Name).
Name	Назва компоненту. Має бути унікальною
Таблиця CpeIdentifiers	
Key	Первинний ключ, отриманий за допомогою функції Sha128(Cpe23Uri).
Cpe23Uri	Ідентифікатор вразливого компоненту згідно загального списку ідентифікаторів CPE [27].
Таблиця Fact_ComponentVulnerabilities	
Id	Сурогатний автогенерований цілочисленний ідентифікатор запису.
Vulnerability_Key	Зовнішній ключ на таблицю Vulnerabilities.
ComponentDefinition_Key	Зовнішній ключ на таблицю ComponentDefinitions.
PublishedDate_Key	Зовнішній ключ на таблицю вимірів Dim_Dates.
ModifiedDate_Key	Зовнішній ключ на таблицю вимірів Dim_Dates.
RemediationDate_Key	Зовнішній ключ на таблицю вимірів Dim_Dates.
Таблиця Dim_Dates	
Key	Первинний ключ, отриманий за допомогою функції Sha128(Date).
Date	Дата.
Year	Рік, до якого належить дата
MonthOfYear	Місяць, до якого належить дата..

Продовження таблиці 2.3

1	2
WeekOfYear	Тиждень, до якого належить дата.
Таблиця ExploitVulnerabilities	
Exploit_Key	Зовнішній ключ на таблицю Exploits.
Vulnerability_Key	Зовнішній ключ на таблицю Vulnerabilities.
Таблиця VulnerabilityCpeIdentifiers	
Vulnerability_Key	Зовнішній ключ на таблицю Vulnerabilities.
CpeIdentifier_Key	Зовнішній ключ на таблицю CpeIdentifiers.
Таблиця ComponentDefinitionCpeIdentifiers	
ComponentDefinition_Key	Зовнішній ключ на таблицю ComponentDefinitions.
CpeIdentifier_Key	Зовнішній ключ на таблицю CpeIdentifiers.
Таблиця Fact_DecisionSimulations	
Id	Сурогатний автогенерований цілочисленний ідентифікатор запису.
Date_Key	Зовнішній ключ на таблицю вимірів Dim_Dates.
ComponentDefinition_Key	Зовнішній ключ на таблицю вимірів ComponentDefinitions. Вказує на компонент, що було обрано згідно стратегії прийняття рішень DecisionStrategy.
DecisionStrategy	Стратегія, що була застосована для обирання найбільш безпечного компонента.
TotalScore	Значення, що було обчислено згідно стратегії.

2.3.3 Розробка ETL-процесу

На рисунку 2.3 зображено процес витягування та трансформації даних для створення сховища даних (DWH).

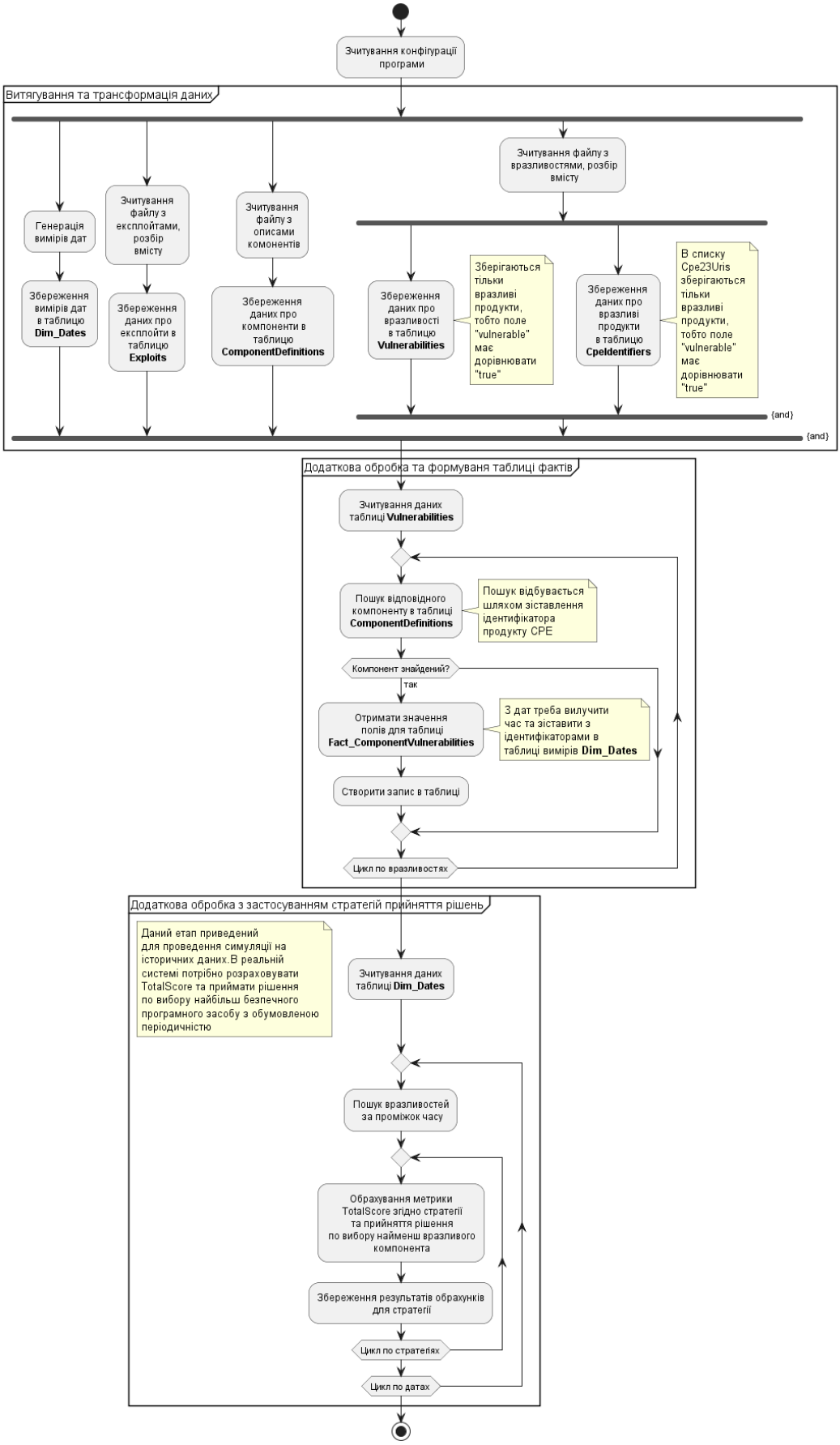

Рисунок 2.3 – Процес трансформації даних для створення сховища даних у вигляді діаграми активності UML

Задля оптимізації швидкості обробки даних застосовано розпаралелювання частин процессу, що відображено на діаграмі відповідними елементами UML.

Процес підготовки даних для аналізу включає дві основні частини:

– витягування та трансформація даних, що передбачає синтаксичний розбір файлів вихідних файлів, отримання даних з них з подальшим збереженням у відповідні таблиці;

– додаткова обробка та формування таблиці фактів, що використовується для побудови візуалізацій та імітаційного моделювання.

2.4 Підготовка даних до статистичного аналізу

2.4.1 Побудова списку ідентифікаторів програмних продуктів

На основі даних з таблиці 2.1 сформуємо список ідентифікаторів вразливих програмних продуктів. Для цього можна скористатися словником [27] або зробити запит по таблиці CpeIdentifiers. Файл формату JSON зі списком ідентифікаторів представлено в додатку Б.

2.4.2 Отримання інформації про дату виправлення вразливості

Нажаль, жоден з ресурсів, що розглянуто в таблиці 2.2 не надає інформацію про дату виправлення тієї чи іншої вразливості у вигляді, придатному для автоматичної обробки. Тому, при моделюванні виправлень вразливостей будемо використовувати припущення, що розробники операційних систем притримуються часових рамок, що викладені в документах NIST, GSA, CISA.

Так, NIST в публікації [28] пропонує притримуватись наступних рамок: від 5 днів для критичних вразливостей до 80 днів для некритичних вразливостей. Також в документі вводиться таке поняття як «важливість ресурсу», від якого також залежить час виправлення.

GSA у своїй настанові з процесу управління вразливостями [29] виставляє наступні рамки:

– 30 днів на виправлення критичних та дуже критичних вразливостей;
– 90 днів на виправлення вразливостей середнього рівня критичності;
– 120 днів на виправлення низьких за критичністю вразливостей.

Американське агентство з кібербезпеки та безпеки інфраструктури [30] наголошує на необхідності дотримання наступних рамок:

– 15 днів на виправлення дуже критичних вразливостей;
– 30 днів на виправлення критичних вразливостей.

Для розбиття на проміжки вразливості за критичністю скористаємося розбиттям, що викладено в документі [26] та сформуємо таблицю 2.4.

Таблиця 2.4 – Час на виправлення вразливості в залежності від рівня критичності

Рівень критичності	**Діапазон рівнів критичності**	**Кількість днів на виправлення**
Низька	0,1 – 3,9	120
Середня	4,0 – 6,9	90
Висока	7,0 – 8,9	30
Критична	9,0 – 10	15

Далі при розрахунках будемо покладатися на «песимистичний» сценарій, тобто, коли вразливість зазначеної критичності виправляється в максимальній строк.

2.5 Застосування інформації про вразливості для підвищення інформаційної безпеки

Операційна система – один з компонентів вебсистеми. При застосуванні крос-платформених технологій розробки програмного забезпечення, операційні системи можуть бути взаємозамінними, що відкриває нові можливості для вирішення проблем безпеки вебзастосунків [18]. Таким чином, можна побудувати систему, що буде мати систему прийняття рішень на основі інформації про вразливості для вибору найбільш безпечної операційної системи у кожний момент часу. Основним питанням тут є те, як розраховувати рівень безпеки окремої операційної системи. Варіантів може бути багато, в даній роботі пропонується розглянути наступні варіанти прийняття рішень:

– найбільш безпечною є та, що має меншу кількість вразливостей у конкретний момент часу;

– найбільш безпечною є та, що має найменшу кількість вразливостей з урахуванням критичності

– найбільш безпечною є та, що має найменшу кількість вразливостей з урахуванням критичності та кількості доступних експлойтів на кожну окрему вразливість.

Для першого випадку формула буде мати наступний вигляд:

$$TotalScore_1 = \sum_{i=m}^{n} f_{count}\ (i), \qquad (2.1)$$

де $f_{count}\ (i)$ – таблична функція, що розраховує кількість вразливостей за день i;

i – порядковий номер дня;

m – початковий номер дня проміжку часу агрегації даних;

n – кінцевий день проміжку часу агрегації даних.

Така модель є швидкою для обчислень, але прирівнює усі вразливості за ступенем критичності. Тож, недоліком такого способу обчислення є

упущення, що кожна вразливість несе свій ступінь небезпеки, який розробники системи оцінки небезпеки CVSS обраховують в [25, 26] та дають загальний числовий показник. Щоб урахувати це, змінимо формулу i), (2.1) та використаємо табличну функцію, що повертає критичність кожної вразливості:

$$f_{cvss}(i) = \sum_k S_k \tag{2.2}$$

де S_k – CVSS значення вразливості k;

k – порядковий номер вразливості дня i.

Таким чином, другий варіант для оцінки безпеки операційної системи буде мати наступний вигляд:

$$TotalScore_2 = \sum_{i=m}^{n} f_{cvss}(i) \tag{2.3}$$

Модель, що наведена у формулі (2.3) враховує критичність кожної вразливості, та все ще не бере до уваги наявність експлойтів. Наявність експлойта для вразливості та їх кількість також має вплив на безпеку, бо експлойт автоматизує атаку, що робить систему більш уразливою. Щоб урахувати експлойти кожної вразливості допрацюємо формулу (2.2) таким чином, щоби враховувати наявність та кількість експлойтів кожної вразливості. При цьому, припустимо, що кожен експлойт збільшує критичність вразливості на розмір критичності самої вразливості. Тоді отримуємо таку формулу:

$$f_{exploit}(i) = \sum_k (S_k + S_k \times E_k) = \sum_k (S_k \times (1 + E_k)) \tag{2.4}$$

де S_k – CVSS значення вразливості k;

E_k – кількість експлойтів вразливості k.

Таким чином, приходимо до третього варіанту розрахунку безпечності операційної системи з урахуванням критичності вразливостей та наявності експлойтів для них:

$$TotalScore_3 = \sum_{i=m}^{n} f_{exploit}\ (i) \tag{2.5}$$

Для прийняття рішення, якою операційною системою користуватися в конкретний момент часу, достатньо обрахувати $TotalScore$ по одній з наведених вище формул для кожної операційної системи та вибрати таку, що має найменше значення вирахуваного показника $TotalScore$.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

3.1 Обґрунтування вибору мови програмування та технологій

При виборі мови програмування увага приділялася тій, що дозволить швидко розробити ETL-рішення. Критеріями вибору мови є наступні:

– ступінь володіння мовою;

– наявність аналітичних бібліотек для вирішення типових задач;

– можливість розробляти крос-платформенні рішення;

– наявність паралелізації та асинхронності, оскільки потрібно буде обробляти великі об’єми даних, то паралелізація може прискорити цей процес.

Таким чином, для розробки модулю ETL була обрана мова C# та платформа .NET8, що нещодавно вийшла в LTS версії [31]. Оскільки модуль повинен виконувати тільки одну функцію – трансформування даних, то, для зручності користування та ефективного розходу ресурсів системи, було обрано застосунок як консольне рішення.

Для ефективної паралельної обробки даних є сенс використовувати готові рішення для платформи .NET. Серед них є платні та безкоштовні. При виборі програмної бібліотеки розглядалися наступні рішення:

– бібліотека ETLBox [32]. Бібліотека має добре задокументоване API, різноманітні конектори, які можна використовувати як джерела даних для трансформування. До недоліків можна віднести відсутність безкоштовного використання бібліотеки в дослідницьких та навчальних цілей без обмежень. На даний час пакет Personal може безоплатно оброблювати до 5000 строк потоку даних;

– бібліотека ChoETL [33]. Має різноманітну кількість конекторів, включаючи CSV, JSON, XML як джерела даних. Має добре задокументоване API з прикладами, до того ж ця бібліотека є безкоштовною для використання. До недоліків можна віднести відсутність інтеграцій з базами даних, ORM EntityFramework тощо;

– бібліотека ETL.Net [34]. Має частково задокументоване API. Пропонує невелику кількість готових конекторів, куди входять EntityFramework та CSV, що нам потрібні для обробки вхідних даних. Для обробки JSON можна використовувати стандартні компоненти платформи .NET з простору імен System.Text.Json [35], тож відсутність конектора на даний час не є блокуючим фактором для використання цієї бібліотеки. Серед переваг можна відмітити можливість безкоштовного використання бібліотеки завдяки ліцензії MIT, під якою ця бібліотека розповсюджується [36]. Таким чином, вибір зупинили на цьому рішенні.

Після зчитування даних з файлів необхідно зберегти отриману інформацію та візуалізувати її. Для простоти зберігання будемо використовувати файлову базу даних SQLite. Для побудови графіків можна застосувати будь-який аналітичний інструмент, що вміє працювати з базою SQLite. Серед альтернатив, що розглядалися, є

– Google Looker (Google Data Studio) [37]. Інструмент підтримує велику кількість джерел даних, велику кількість візуалізацій. Нажаль, цей інструмент не працює з SQLite. Має добре описану документацію.

– Metabase [38]. Має як платні так і безкоштовні плани. При запуску на власному обладнанні є безкоштовним, а обмеження безкоштовної версії не впливають на функціональність при персональному використанні. Потребує базу PostgreSQL для своєї роботи та може запускатися в контейнерах Docker. Має добре описану документацію.

– Apache Superset [39]. Є повністю безкоштовним інструментом, має велику кількість візуалізацій, може інтегруватися з SQLite. Має детально описану документацію.

У статі [40] порівнюються такі інструменти для візуалізації як Metabase, Redash, Apache Superset. В нашому випадку вибір зроблено на користь Metabase з точки зору зручності розгортання з контейнеру Docker та зручності налаштування.

3.2 Програмна реалізація

Налаштування програми винесемо у конфігураційний файл JSON, для роботи з якими у Microsoft є готове рішення, що міститься у просторі імен Microsoft.Extensions.Configuration.*. Рисунок 3.1 відображує діаграму класів, яка використовується для доступу до конфігураційних секцій застосунку.

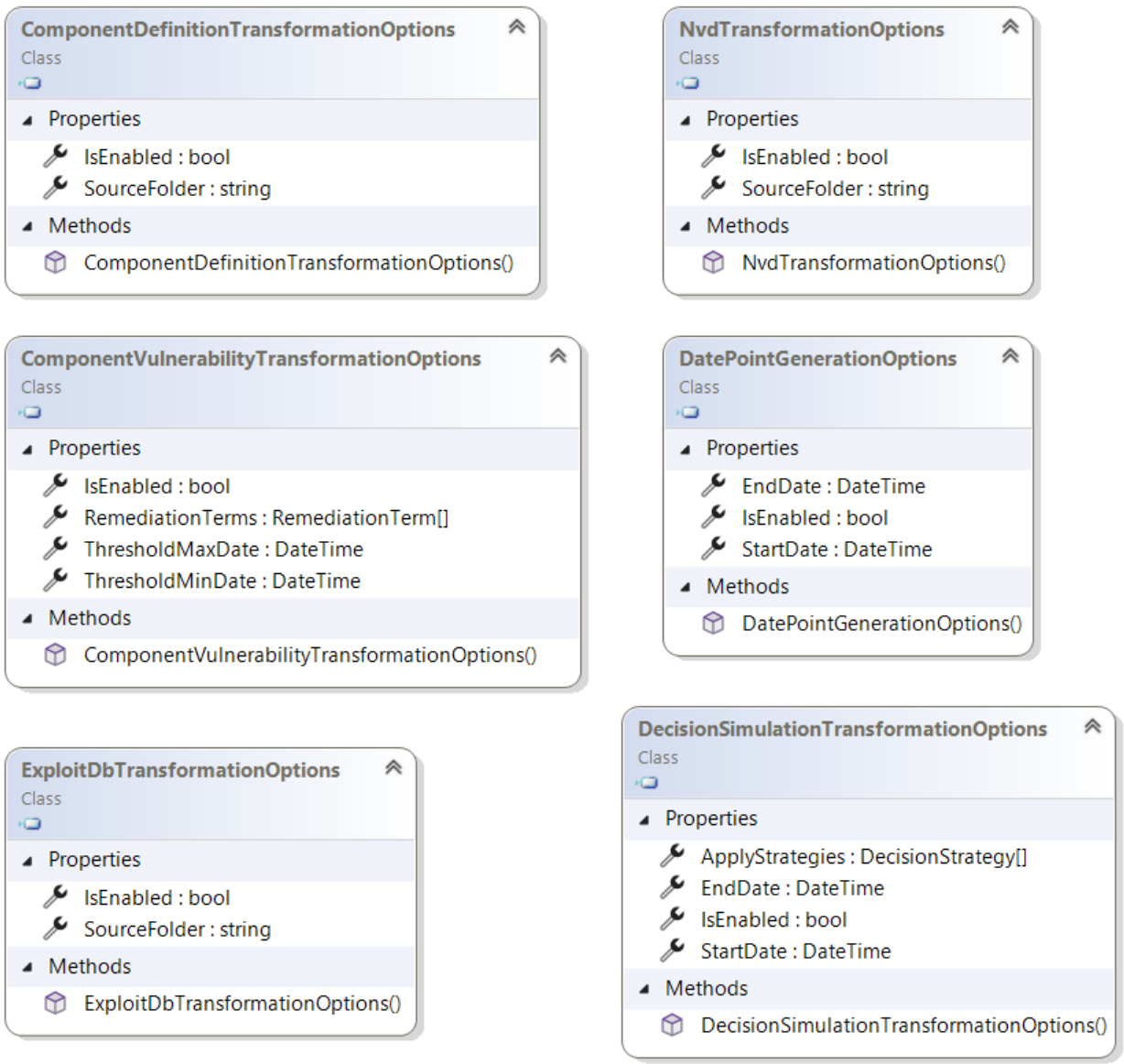

Рисунок 3.1 – Діаграма класів для конфігурації трансформацій

Для виконання симуляцій та можливістю підключення нових реалізацій для системи прийняття рішень, скористаємося патерном «стратегія» та розробимо класи, що реалізовують стратегії, описані у розділі 2.5. Дані стратегії будуть мати один загальний метод – Apply. Цей метод відповідальний за виконання розрахунків TotalScore для конкретної стратегії та за вибір найбільш безпечної операційної системи. Таким чином, можна виділити загальний інтерфейс IDecisionStrategy, що буде визначати базовий набір методів для усіх стратегій. Результати своїх розрахунків та прийняте рішення дані класи записують у таблицю Fact_DecisionSimulations. Рисунок 3.2 відображує діаграму класів для реалізації стратегій.

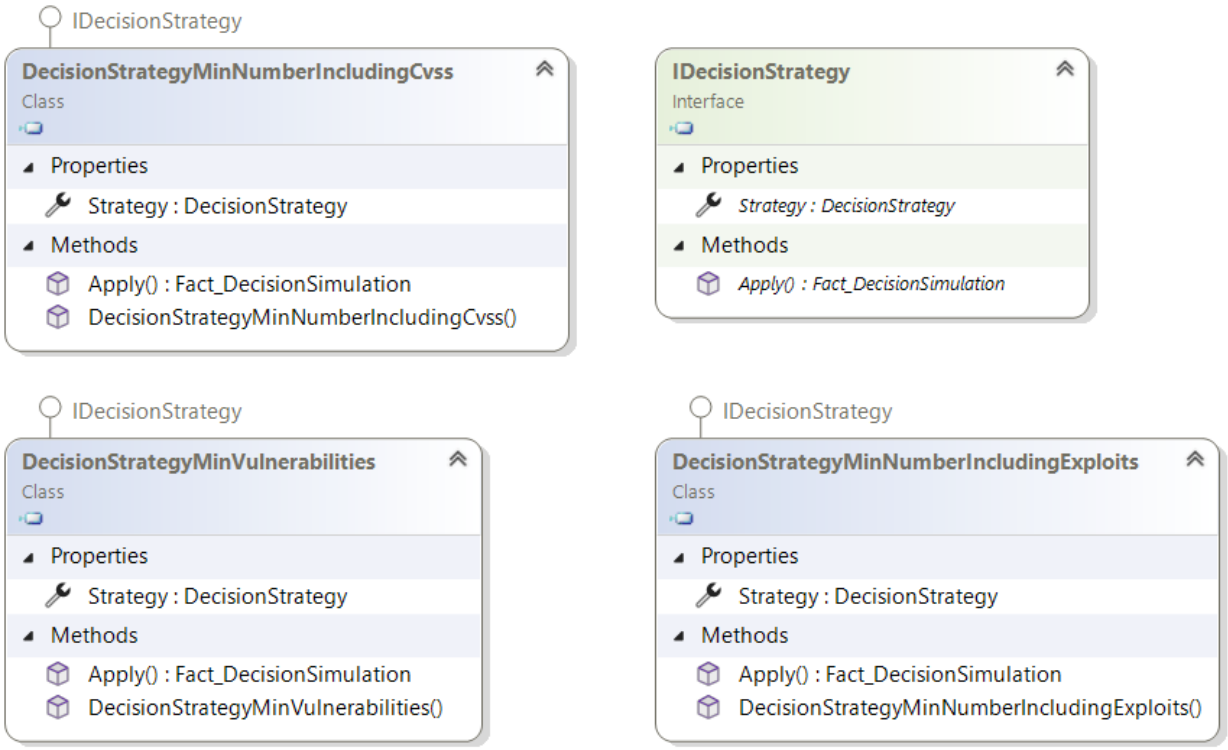

Рисунок 3.2 – Діаграма класів, відповідальних за стратегію вибору найбільш безпечної операційної системи

Принцип роботи бібліотеки ETL.Net побудований на таких поняттях як вузол (node), потік (stream) та оператор (operator) [41]. Рисунок 3.3 відображає схему обробки даних, що реалізована в бібліотеці.

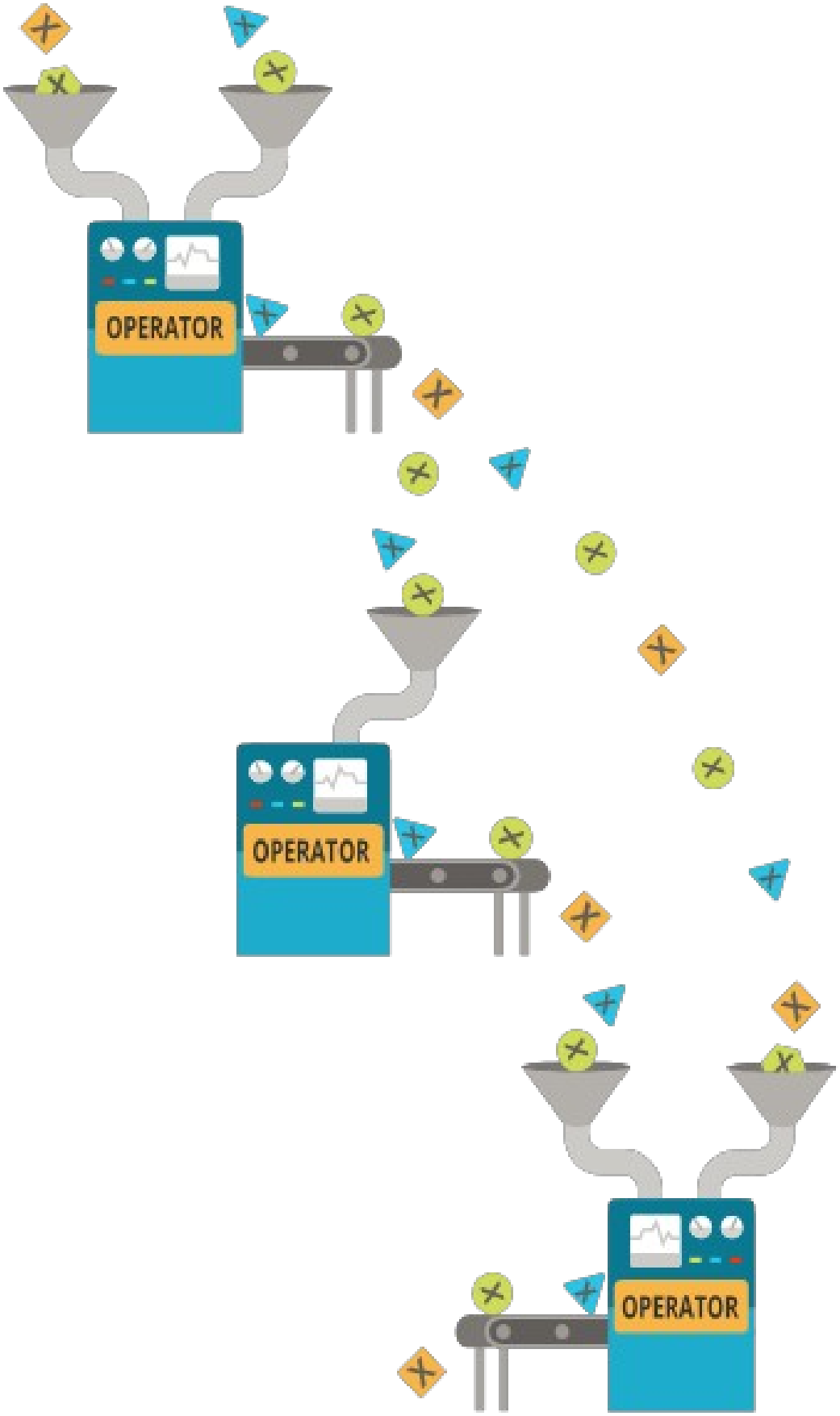

Рисунок 3.3 – Процес трансформації даних в бібліотеці ETL.NET

На рисунку 3.4 зображена діаграма класів для виконання ETL трансформацій, яка включає:

- NvdJsonToDatabaseTransformation. Клас для трансформації JSON-файлів бази даних. Самі файли база NVD постачає в виді архівів ZIP. Даний клас сканує усі ZIP файли, зазначені в конфігурації, розпаковує архів та обробляє JSON файл, що міститься в архіві;
- ComponentDefinitionJsonToDatabaseTransformation. Клас виконує трансформацію опису компоненту з JSON файлу;
- ExploitsCsvToDatabaseTransformer. Клас для трансформації CSV-файлів бази ExploitDB. Даний клас сканує CSV файли в зазначеній директорії та виконує їх трансформацію;
- DatePointGeneration. Клас відповідальний за генерацію вимірів дат та наповнення таблиці Dim_Dates;
- ComponentVulnerabilityTransformation. Клас відповідальний за заповнення таблиці фактів Fact_ComponentVulnerabilities. При трансформації даний клас при трансформації використовує уже збережені в таблицях дані про вразливості, експлойти, компоненти, ідентифікатори CPE та зберігає результати своєї роботи в таблицю Fact_ComponentVulnerabilites;
- DecisionSimulationTransformation. Клас відповідальний за симуляцію прийняття рішень про вибір найбільш безпечної операційної системи. При трансформації клас використовує інформацію про вразливості, експлойти та критичність вразливостей, щоб обрахувати параметр TotalScore та обрати найбільш безпечну операційну систему в кожен момент часу.

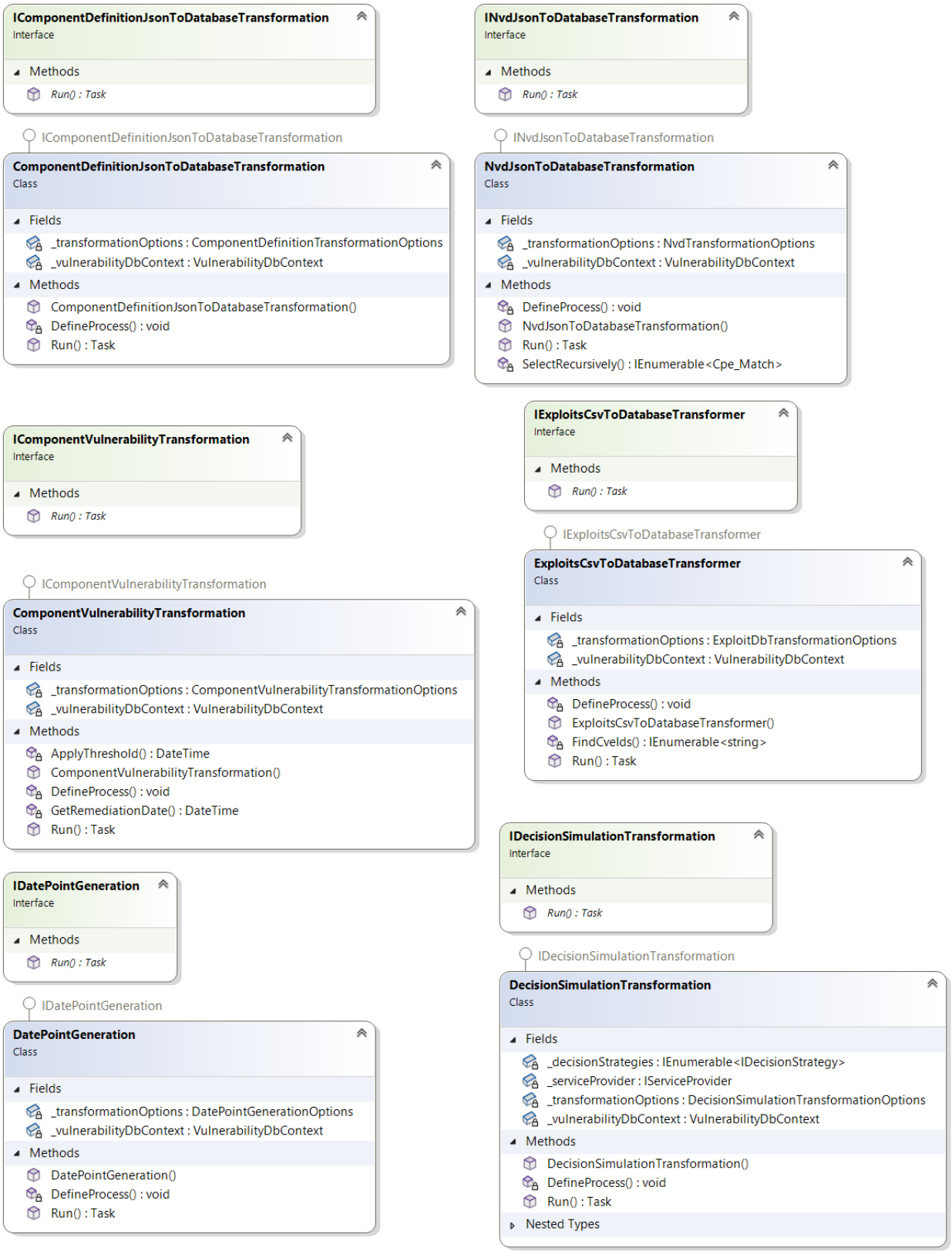

Рисунок 3.4 – Діаграма класів для ETL-трансформацій

Для доступу до даних у базі даних SQLite будемо використовувати ORM EntityFramework. Він дозволяє розробнику сконцентруватися на даних, а решту рутинних операцій ORM бере на себе, серед яких:

– створення скриптів міграцій для зміни схеми даних сховища;

– автоматичний мапінг даних з бази даних на об’єкти C#;

– абстрагування від конкретного движка бази даних, надаючи абстракцію у вигляді LINQ та DbContext;

– контроль за відповідністю моделі бази даних до моделі класів-сутностей у застосунку;

– відстежування змін в даних та автоматичне генерування відповідних запитів на вставку (INSERT), оновлення (UPDATE) та видалення даних (DELETE);

– трансляція LINQ запитів у конкретні запити (SELECT) на мові SQL для конкретного движка бази даних;

– та ін.

Рисунок 3.5 зображує діаграму класів-сутностей для маніпулювання даними в базі даних SQLite за допомогою EntityFramework.

На рисунку 3.6 зображена діаграма класів, необхідних для функціонування та початкової ініціалізації застосунка, що містить наступні класи:

– Program. Виконує базове налаштування застосунку шляхом виклику методів класу Startup в такій послідовності: ConfigureServices, Configure, Run;

– ApplicationOptions. Містить початкові налаштування застосунку;

– Startup. Клас, що проводить ініціалізацію залежностей та конфігурування ORM EntityFramework;

– KeyUtils. Містить меттод ComputeKey для обрахування криптографичної хеш-функції Sha128. Використовується для генерації первинних ключів для сутностей;

– RegisterServices. Реєструє класи для ETL трансформацій для їх подальшого запуску.

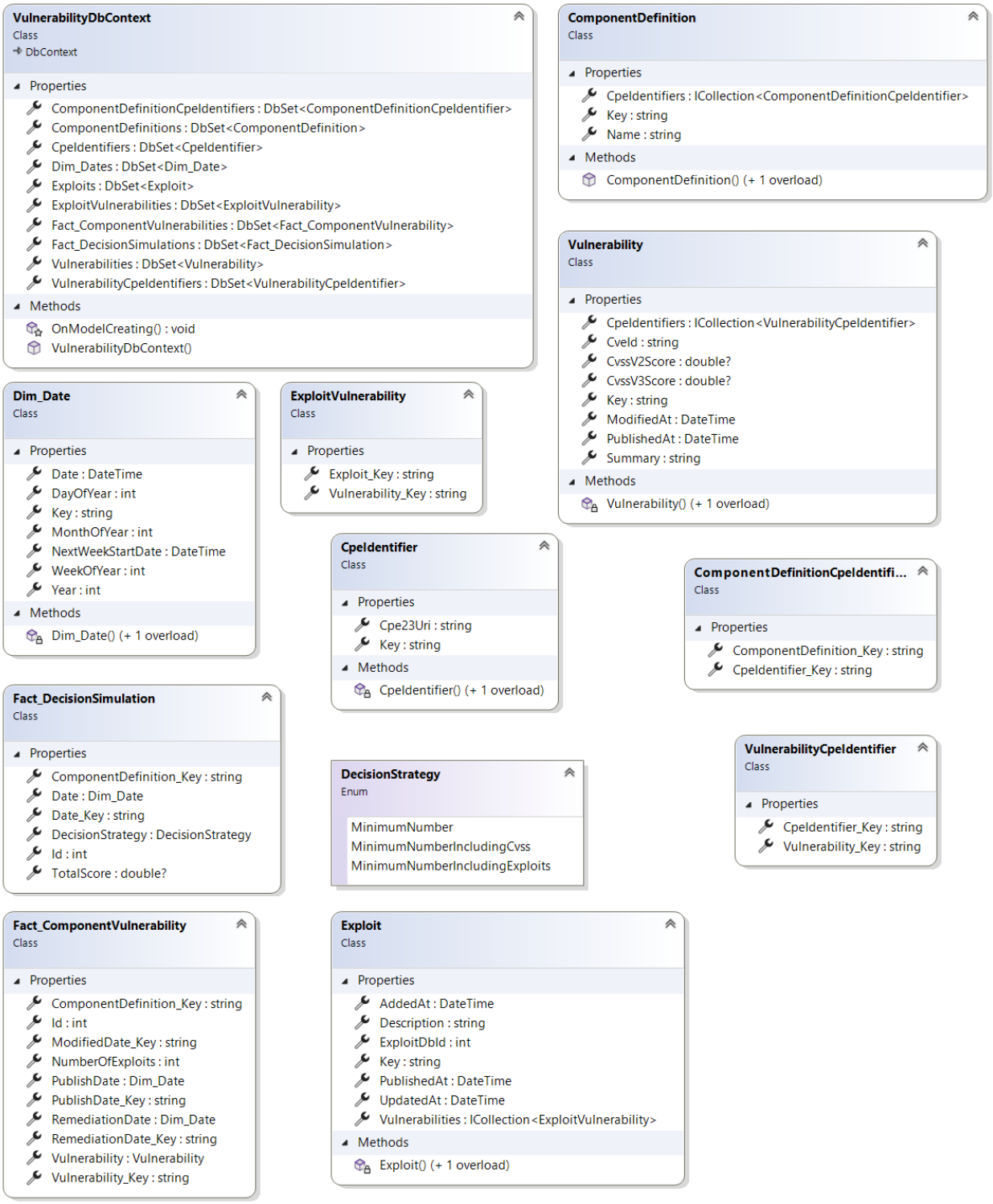

Рисунок 3.5 – Діаграма класів-сутностей EntityFramework та контексту VulnerabilityDbContext для доступу к даним у базі даних

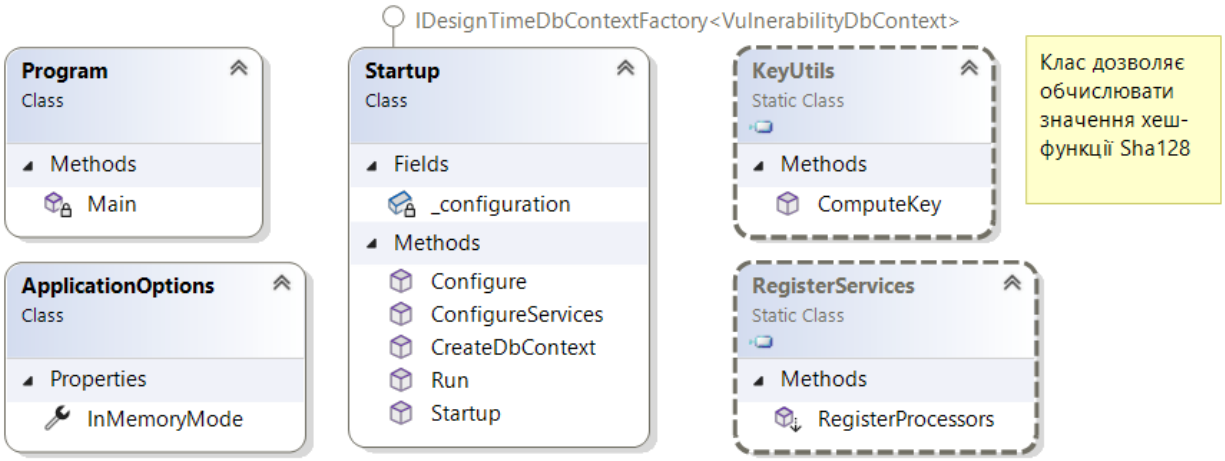

Рисунок 3.6 – Діаграма утилітарних класів, що необхідні для функціонування та початкової ініціалізації застосунка

Для запуску Metabase сформуємо файл для утиліти docker-compose, що наведено у додатку А та виконаємо у консолі наступну команду для локального розгортання Metabase з PostgreSQL на базі образу Linux Alpine:

docker-compose -f .\metabase-postgres.docker-compose.yml up

Результатом виконання цієї команди буде запуск двох контейнерів, що ми можемо побачити в інтерфейсі застосунку Docker Desktop. Рисунок 3.7 відображає зовнішній вид вікна застосунку Docker Desktop після запуску Metabase у контейнері.

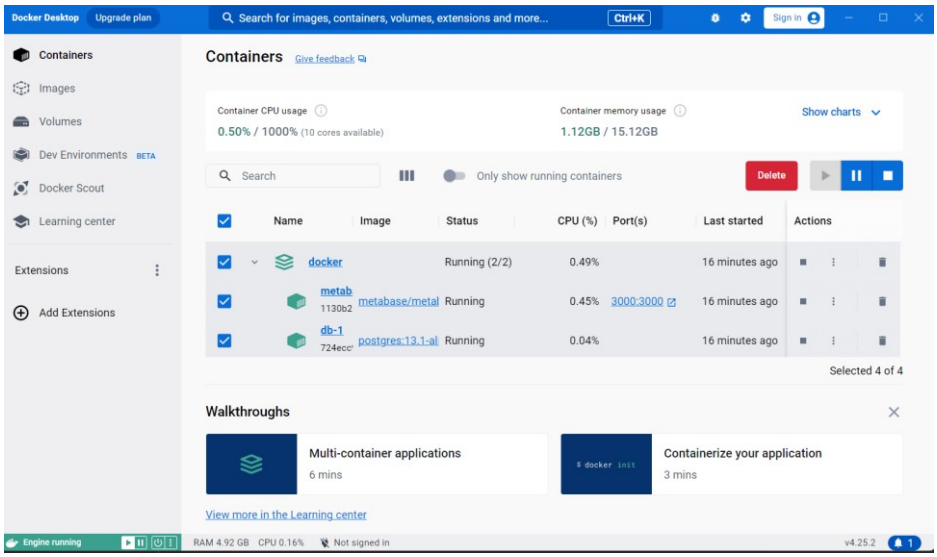

Рисунок 3.7 – Результат запуску Metabase та PostgreSQL у контейнері

3.3 Інструкція користувача

3.2.1 Запуск процесу трансформації

Для запуску процесу трансформації достатньо виконати наступну команду з консолі:

.\VulnerabilityETL.exe

Результатом виконання команди буде зчитування файлу appsettings.json з налаштуваннями та запуск процесу трансформації.

Для керування утилітою в файлу конфігурації передбачені секції, що розглянуто нижче.

3.2.2 Секція ConnectionStrings

Дозволяє задавати строки підключення до бази SQLite. Містить такі ключі:

– VulnerabilityDb – містить строку підключення до файлової бази даних. Приклад: *Data Source=.\\vulnerabilitydb.sqlite3;* ;

– VulnerabilityDbInMemory – містить строку підключення для роботи з SQLite в режимі «тільки в пам’яті». Після перезапуску утиліти усі дані будуть втрачені. Приклад: *:memory:;Version=3;New=True;* .

3.2.3 Секція ApplicationOptions

Дозволяє керувати опціями запуску. Містить параметр InMemoryMode. Можливі значення:

– *true*. Буде використана строка підключення VulnerabilityDbInMemory при роботі з базою;

– *false*. Буде використана строка підключення VulnerabilityDb.

3.2.4 Секція ExploitDbTransformationOptions

Дозволяє керувати опціями трансформації файлу бази ExploitDB. Має наступні параметри:

– IsEnabled. Якщо дорівнює false, то ця трансформація не виконується;

– SourceFolder. Містить шлях до директорії, де зберігаються CSV-файли для обробки. Приклад: .*Data**Exploits*\\ .

3.2.5 Секція NvdTransformationOptions

Дозволяє керувати опціями трансформації файлу бази NVD. Має наступні параметри:

– IsEnabled. Якщо дорівнює false, то ця трансформація не виконується;

– SourceFolder. Містить шлях до директорії, де зберігаються Zip-файли для обробки. Приклад: .*Data**Nvd*\\ .

3.2.6 Секція ComponentDefinitionTransformationOptions

Дозволяє керувати опціями трансформації для файлу з визначенням компонентів. Має наступні параметри:

– IsEnabled. Якщо дорівнює false, то ця трансформація не виконується;

– SourceFolder. Містить шлях до директорії, де зберігаються JSON-файли для обробки. Приклад: .*Data**ComponentDefinitions*\\ .

3.2.7 Секція ComponentVulnerabilityTransformationOptions

Дозволяє керувати опціями трансформації та наповнення таблиці фактів Fact_ComponentVulnerabilities. Має наступні параметри:

– IsEnabled. Якщо дорівнює *false*, то ця трансформація не виконується;

– ThresholdMinDate. Початкова дата періоду, за який розглядаються дані;

– ThresholdMinDate. Кінцева дата періоду, за який розглядаються дані;

– RemediationTerms. Містить співвідношення критичності вразливості до терміну усунення вразливості. Використовується при імітаційному моделюванні процесів усунення вразливостей. Містить ScoreMin та ScoreMax для зазначення проміжку критичності оцінки, на якій застосовується правило, та TermDays для зазначення кількості днів для усунення вразливості. Рисунок 3.8 відображає приклад налаштування секції.

```
"ComponentVulnerabilityTransformationOptions": {
  "IsEnabled": true,
  "ThresholdMinDate": "2008-01-01",
  "ThresholdMaxDate": "2030-01-01",
  "RemediationTerms": [
    {
      "ScoreMin": 0.1,
      "ScoreMax": 3.9,
      "TermDays": 120
    },
    {
      "ScoreMin": 4,
      "ScoreMax": 6.9,
      "TermDays": 90
    },
    {
      "ScoreMin": 7,
      "ScoreMax": 8.9,
      "TermDays": 30
    },
    {
      "ScoreMin": 9,
      "ScoreMax": 10,
      "TermDays": 15
    }
  ]
},
```

Рисунок 3.8 – Приклад налаштування секції ComponentVulnerabilityTransformationOptions

3.2.8 Секція DatePointGenerationOptions

Дозволяє керувати опціями заповнення таблиці Dim_Dates. Має наступні параметри:

– IsEnabled. Якщо дорівнює *false*, то ця трансформація не виконується;

– StartDate. Визначає початкову дату для генерації значень для таблиці вимірів дат Dim_Dates. Приклад: *2008-01-01*.

– EndDate. Визначає кінцеву дату для генерації значень для таблиці вимірів дат Dim_Dates. Приклад: *2008-01-01*.

3.2.9 Секція DecisionSimulationTransformationOptions

Дозволяє керувати опціями заповнення таблиці Fact_DecisionSimulation. Має наступні параметри:

– IsEnabled. Якщо дорівнює *false*, то ця трансформація не виконується;

– StartDate. Визначає початкову дату симуляції вибору найбільш безпечної операційної системи. Приклад: *2008-01-01* ;

– EndDate. Визначає кінцеву дату симуляції вибору найбільш безпечної операційної системи. Приклад: *2023-11-01* ;

– ApplyStrategies. Визначає список стратегій, який використовувати при симуляції. «1» дорівнює стратегії вибору операційної системи з найменшою кількістю вразливостей. «2» дорівнює стратегії вибору операційної системи з найменшою кількістю вразливостей за критичністю. «3» враховує критичність та кількість експлойтів для вразливості при виборі найбільш безпечної.

3.4 Тестування розробленої програми та імітаційне моделювання

Таблиця 3.1 містить набір тестів для перевірки застосунку.

Таблиця 3.1 – Тестові набори для перевірки застосунку

№	Функціональність, що перевіряється	Кроки перевірки	Результат
1	**2**	**3**	**4**
1	Генерація дат Dim_Dates	1. Включити в налаштуваннях застосунку генерацію дат Dim_Dates 2. Вказати період генерації дат в налаштуваннях.	Успіх
		3. Запустити застосунок Очікуваний результат: після завершення трансформації таблиця Dim_Dates заповнена даними з датами за вказаний період. Дані відповідають налаштуванням.	
2	Зчитування файлу JSON з описом компонентів	1. Налаштувати секцію трансформації файлу зі списком компонентів. Вказати директорію з цільовими файлами 2. Включити секцію в налаштуваннях. 3. Запустити додаток. Очікуваний результат: після завершення роботи додатку таблиця ComponentDefinitions заповнена даними з вихідного файлу. Дані	Успіх

Продовження таблиці 3.1

1	2	3	4
		відповідають вмісту файлів, що були завантажені.	
3	Зчитування файлу CSV з даними ExploitDb	1. Налаштувати секцію трансформації файлів бази ExploitDB. Вказати директорію з цільовими файлами. 2. Включити секцію в налаштуваннях. 3. Запустити додаток. Очікуваний результат: після завершення роботи додатку таблиця	Успіх
		Exploits заповнена даними з вихідного файлу. Дані відповідають вмісту файлів, що були завантажені.	
4	Зчитування файлу JSON з даними ExploitDb	1. Налаштувати секцію трансформації файлів бази NVD. Вказати директорію з цільовими файлами. 2. Включити секцію в налаштуваннях. 3. Запустити додаток. Очікуваний результат: після завершення роботи додатку таблиця Vulnerabilities заповнена даними з вихідного файлу. Дані відповідають вмісту файлів, що були завантажені.	Успіх
5	Виконання додаткової обробки з заповненням	1. Налаштувати секцію для заповнення таблиці фактів Fact_ComponentVulnerabilities. Вказати строки виправлення	Успіх

Продовження таблиці 3.1

1	2	3	4
	таблиці фактів вразливостей.	вразливостей в залежності від критичності. 2. Включити секцію в налаштуваннях. 3. Запустити додаток. Очікуваний результат: після завершення роботи додатку таблиця Fact_ComponentVulnerabilities заповнена даними з вихідного файлу. Кількість експлойтів відповідає даним в таблиці ExploitVulnerabilities.	
6	Виконання додаткової обробки з заповненням таблиці фактів прийнятих рішень симуляцій.	1. Налаштувати секцію для заповнення таблиці фактів Fact_DecisionSimulations. Вказати строки виправлення вразливостей в залежності від критичності. 2. Включити секцію в налаштуваннях. 3. Запустити додаток. Очікуваний результат: після завершення роботи додатку таблиця Fact_DecisionSimulations заповнена даними з вихідного файлу. Результати симуляцій відповідають вихідним даним в таблицях Fact_ComponentVulnerabilities.	Успіх

3.3 Ілюстрація роботи системи

Розроблений додаток дозволяє завантажувати дані та трансформувати їх для подальшої візуалізації. Приклади екранів, що генерує система наведено на рисунках

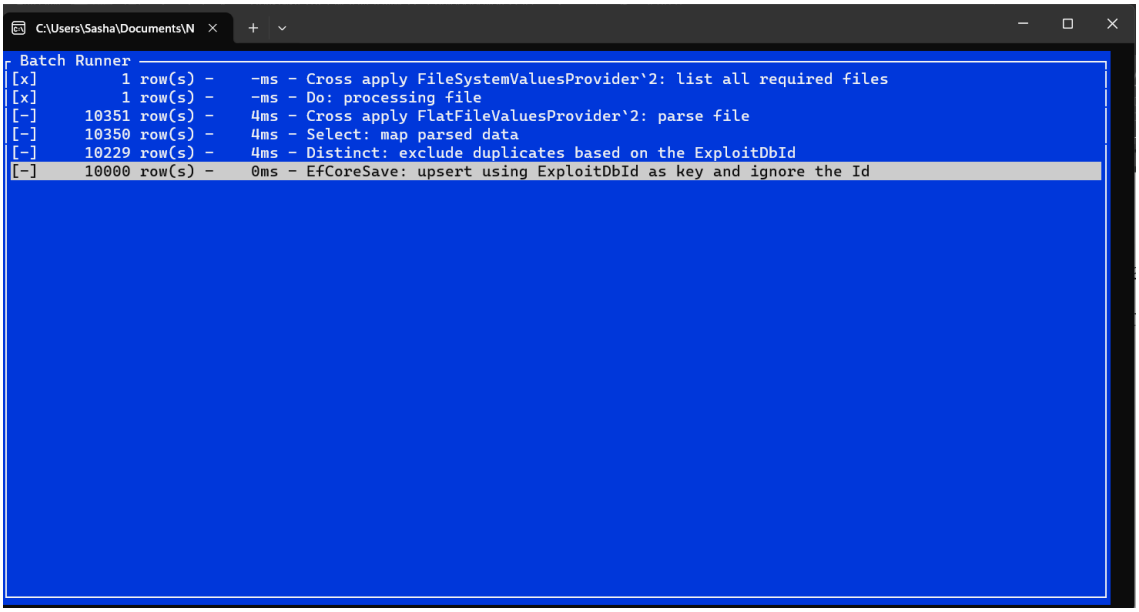

Рисунок 3.9 – Зовнішній вигляд вікна застосунку на етапі трансформування CSV файлу з експлойтами

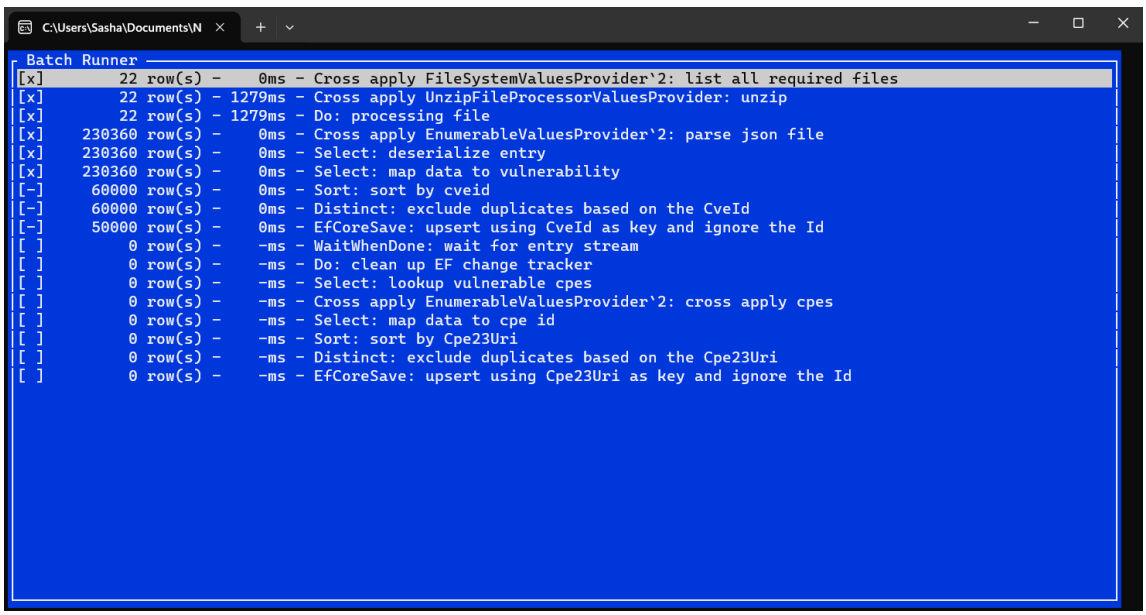

Рисунок 3.10 – Зовнішній вигляд вікна застосунку на етапі обробки JSON файлів бази NVD

Отримані трансформовані дані дозволяють отримати візуалізацію за допомогою інструменту Metabase або іншого аналітичного інструменту. Приклади візуалізацій наведені на рисунках 3.11 – 3.16.

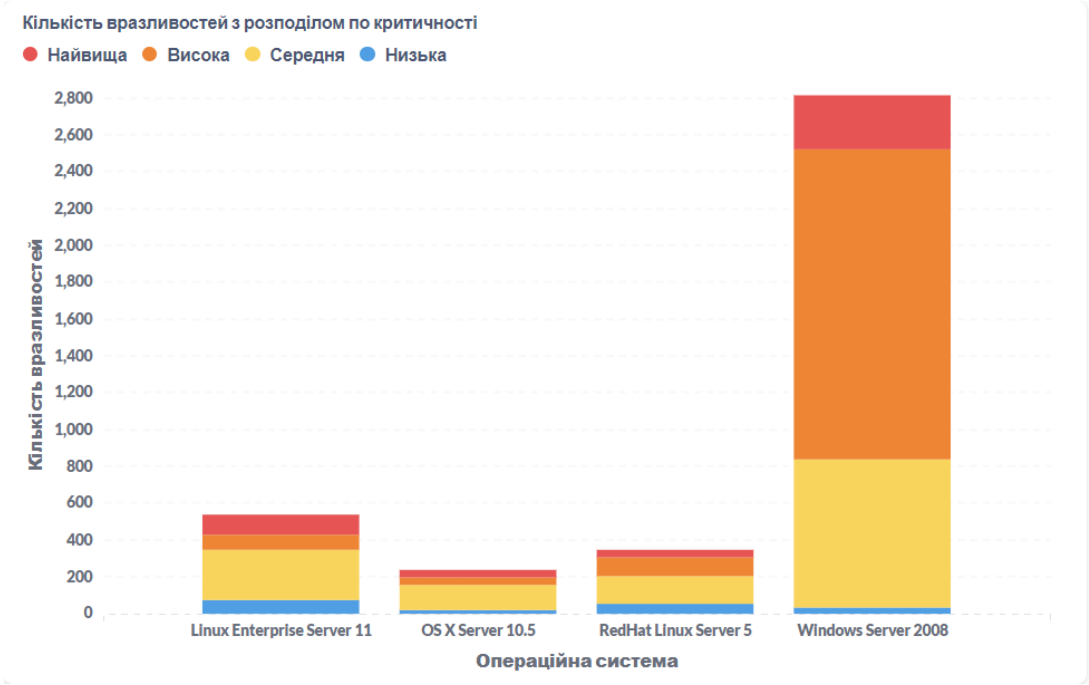

Рисунок 3.11 – Розподіл кількості вразливостей за їх критичністю по операційних системах

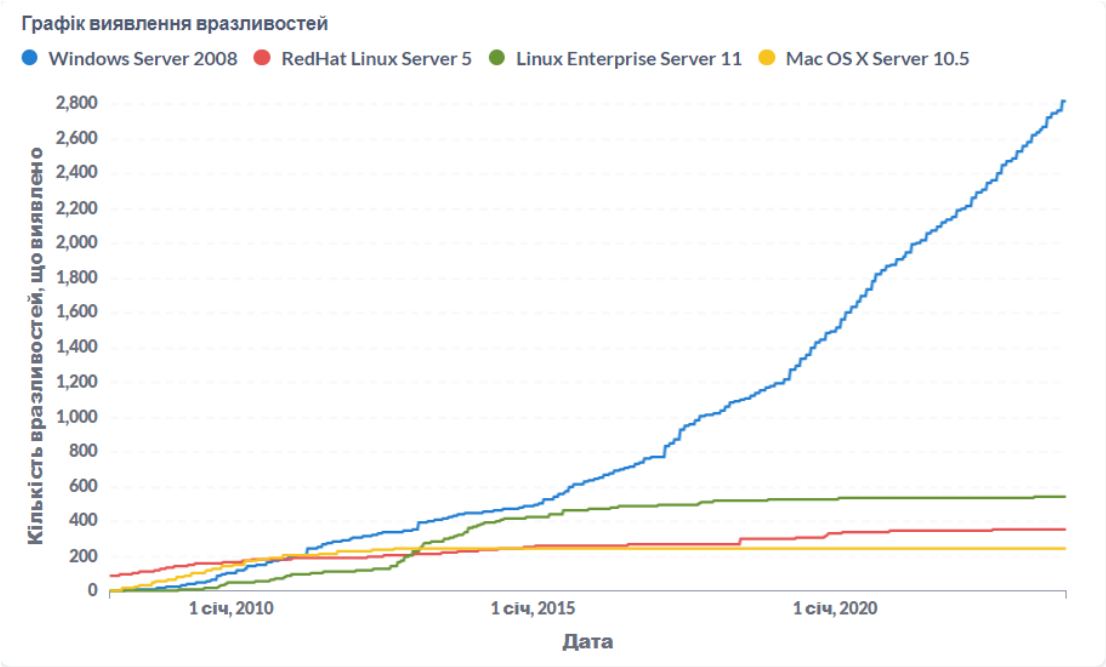

Рисунок 3.12 – Сумарна кількість виявлених вразливостей за часом по операційних системах

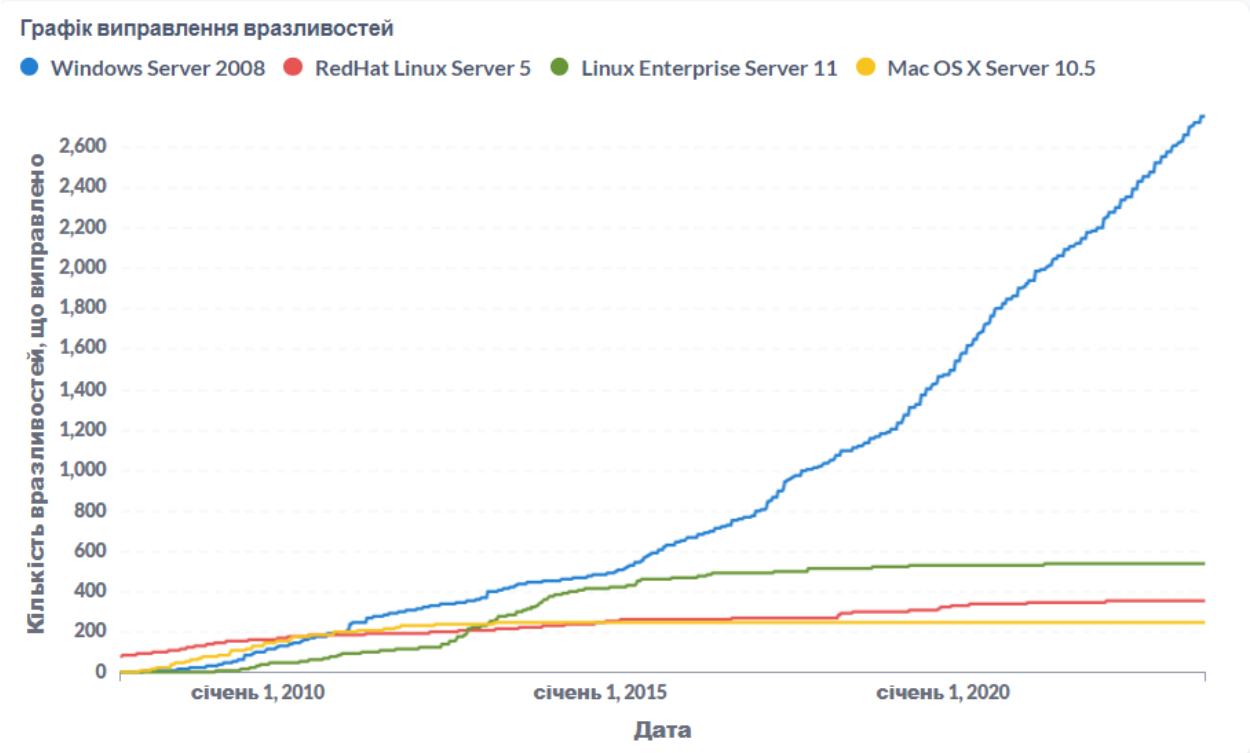

Рисунок 3.13 – Сумарна кількість виправлених вразливостей за часом по операційних системах

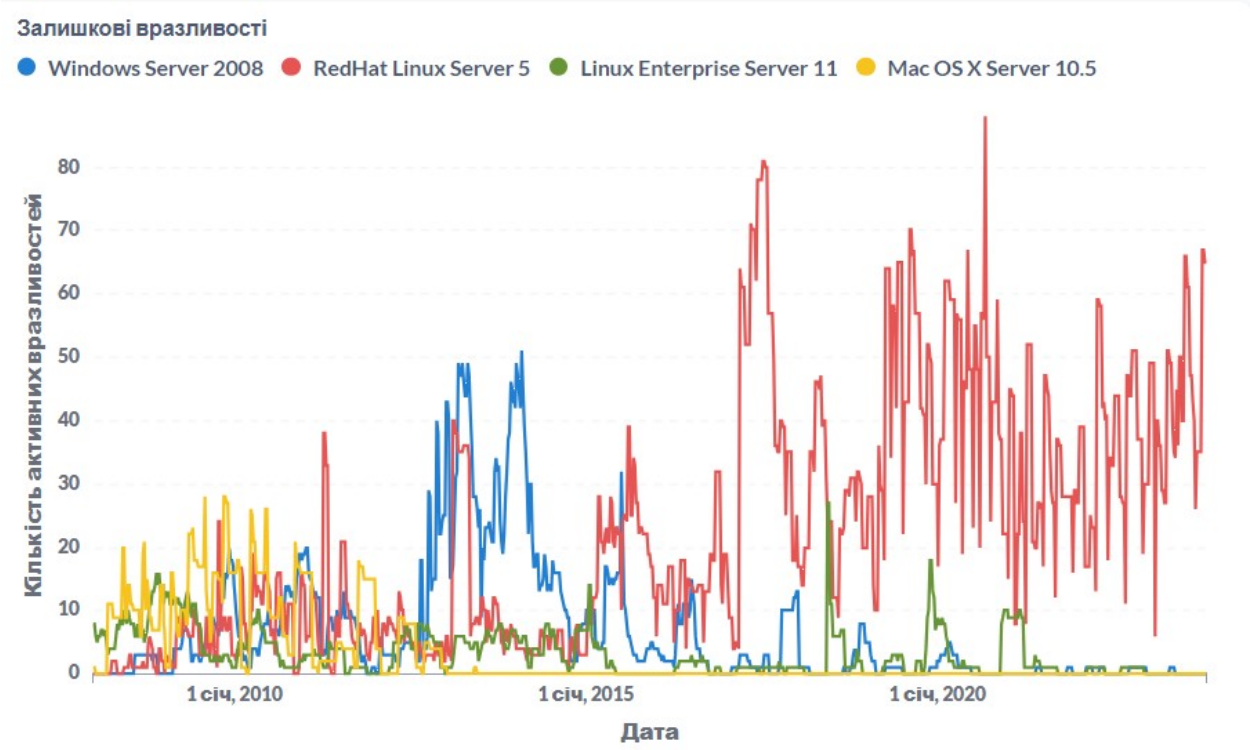

Рисунок 3.14 – Кількість вразливостей, що залишаються невиправленим за часом по операційних системах

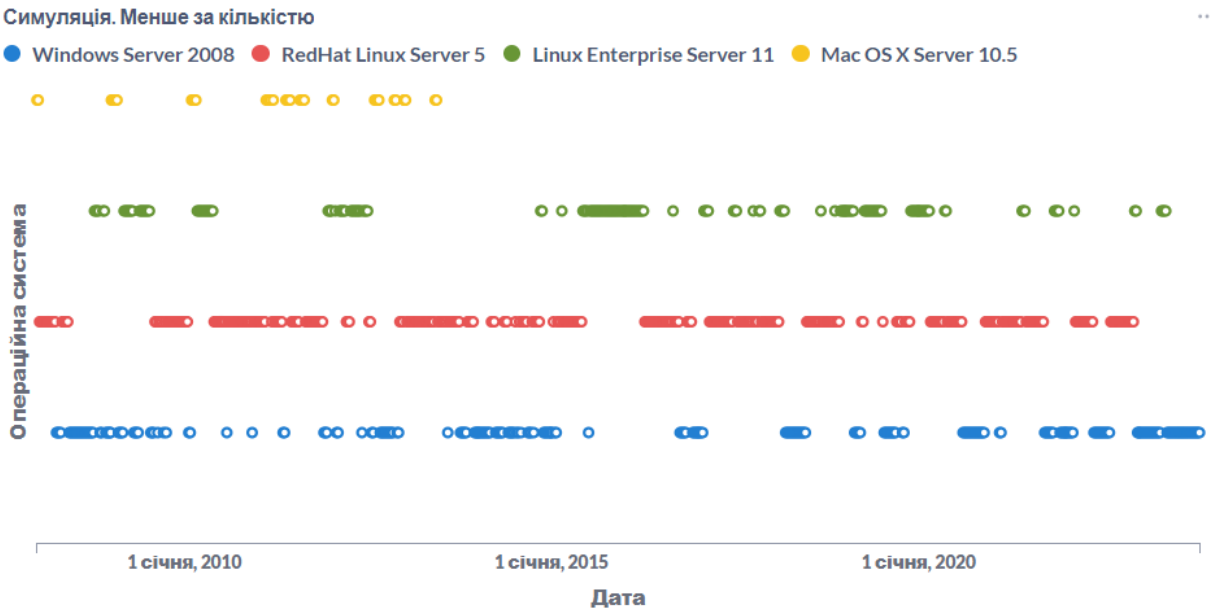

Рисунок 3.15 – Симуляція вибору операційної системи, яка містить найменшу кількість вразливостей відповідно до формули (2.1)

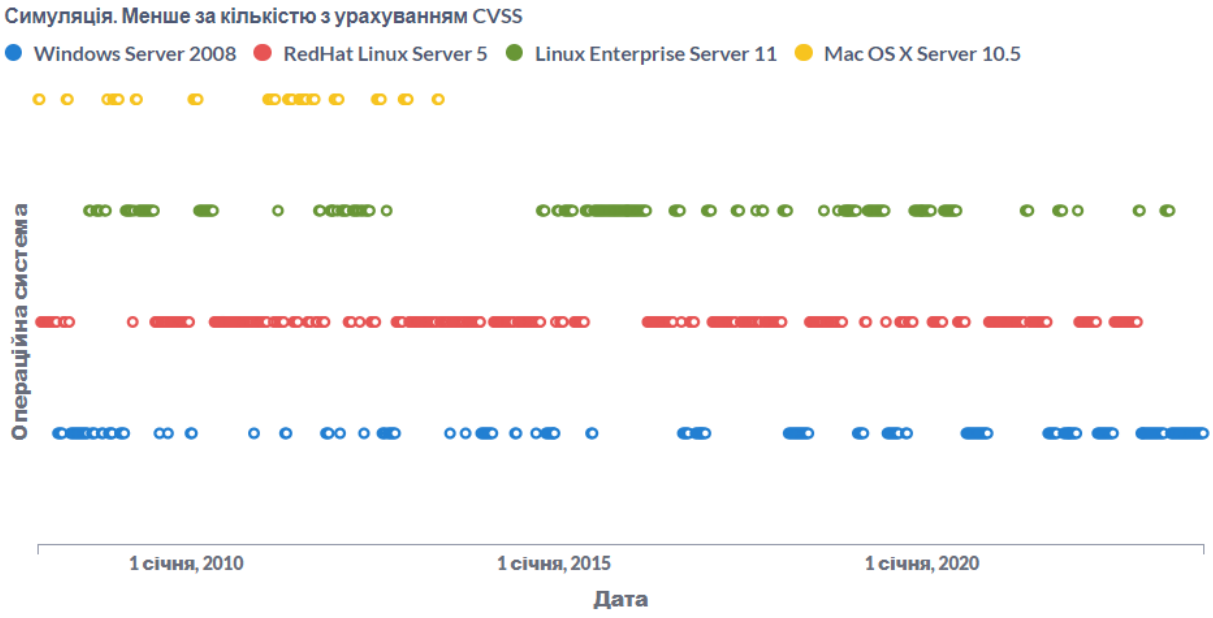

Рисунок 3.16 – Симуляція вибору операційної системи, яка містить найменшу кількість вразливостей з урахуванням критичності кожної вразливості відповідно до формули (2.3)

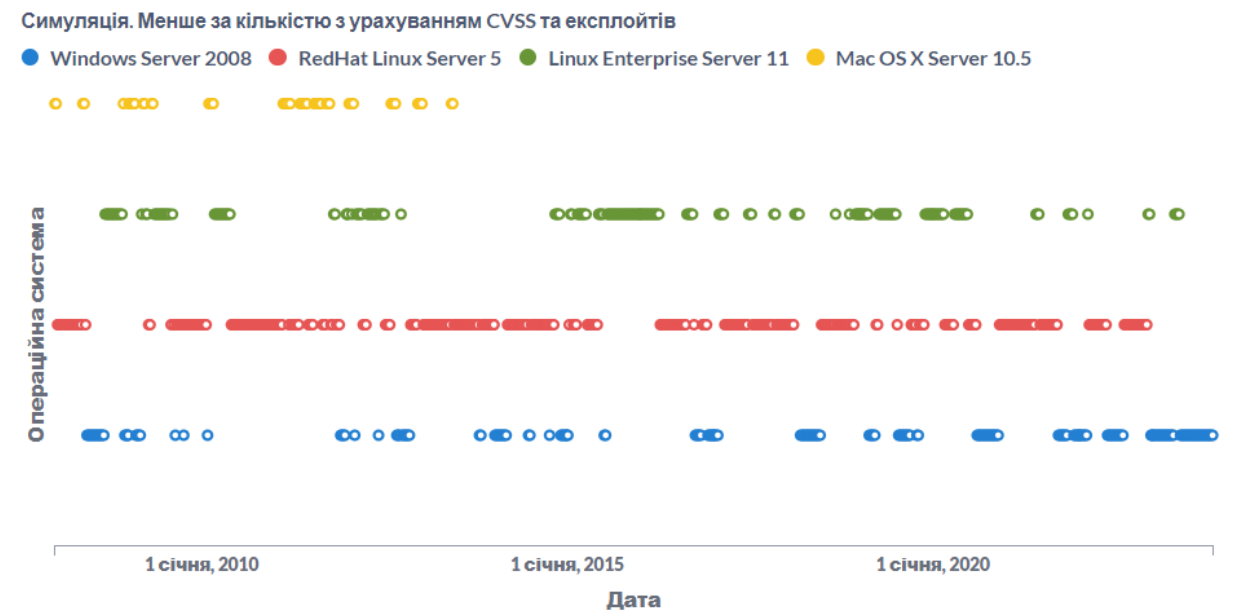

Рисунок 3.17 – Симуляція вибору операційної системи, яка містить найменшу кількість вразливостей з урахуванням критичності кожної вразливості та наявності експлойтів для цих вразливостей відповідно до формули (2.5)

Рисунок 3.18 – Обраховане значення TotalScore відповідно до формул (2.1), (2.3), (2.5)

Також одним результатів дослідження є створення власної бази даних вразливостей та експлойтів, що може бути використана в подальших

дослідженнях. Таблиця 3.2 демонструє приклади вразливостей, що мають більше одного експлойта.

Таблиця 3.2 – Приклади вразливостей, що мають більше одного експлойта

Операційна система	**Кількість експлойтів**	**Вразливість**
RedHat Linux Server 5	11	CVE-2000-0844
Windows Server 2008	9	CVE-2014-6332
Windows Server 2008	6	CVE-2009-3103
Windows Server 2008	6	CVE-2014-4114
Windows Server 2008	6	CVE-2008-4250
Windows Server 2008	6	CVE-2014-6352
Linux Enterprise Server 11	5	CVE-2009-3547
RedHat Linux Server 5	5	CVE-2016-5195
Linux Enterprise Server 11	4	CVE-2014-0038
Windows Server 2008	4	CVE-2014-4113
Windows Server 2008	4	CVE-2016-0099
Windows Server 2008	4	CVE-2019-0708
Windows Server 2008	3	CVE-2010-3147
RedHat Linux Server 5	3	CVE-2007-5962
Windows Server 2008	3	CVE-2008-4037
Windows Server 2008	3	CVE-2010-0480
Linux Enterprise Server 11	3	CVE-2009-1185
Linux Enterprise Server 11	3	CVE-2017-1000366
Windows Server 2008	3	CVE-2013-3660
Windows Server 2008	3	CVE-2012-0217
Windows Server 2008	3	CVE-2013-3661
Windows Server 2008	3	CVE-2016-7255
Windows Server 2008	3	CVE-2017-0199

ВИСНОВКИ

У кваліфікаційній роботі був запропонований метод для аналізу вразливостей серверних операційних систем на основі даних з відкритих джерел, а також зроблено імітаційне моделювання системи, що працює з у кожний проміжок таким чином, щоби використовувати найбільш безпечні компоненти з точки зору можливих атак на вразливості.

У ході виконання роботи отримані наступні результати:

– проаналізовані джерела даних про вразливості;

– розроблено метод для збору інформації про вразливості та експлойти з відкритих джерел;

– розроблено модель сховища даних DWH для виконання подальших аналітичних запитів;

– розроблено ETL-процес для трансформування даних з відкритих джерел з подальшим їх зберіганням у сховищі даних;

– розроблені аналітичні формули для системи прийняття рішень по вибору найбільш безпечної операційної системи;

– проведена симуляція на історичних даних для системи прийняття рішень.

Результати дослідження апробовано у вигляді тез доповідей під час II Міжнародної науково-практичної конференції «Young scientists and methods of improving modern theories» [1].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Білобородов, О. (2023, September). Дослідження вразливостей серверних операційних систем. In The 2nd International scientific and practical conference “Young scientists and methods of improving modern theories”(September 26–29, 2023) Milan, Italy. International Science Group. 2023. 196 p. (p. 173).

2. Баранов, О. А. (2014). Про тлумачення та визначення поняття «кібербезпека». Правова інформатика, 2(42), 54-62.

3. Avizienis, Algirdas, and J-C. Laprie. "Dependable computing: From concepts to design diversity." Proceedings of the IEEE 74.5 (1986): 629-638.

4. 15th Annual CSI/FBI Computer Crime and Security Survey. URL: https://cours.etsmtl.ca/gti619/documents/divers/CSIsurvey2010.pdf (дата звернення 05.11.2023).

5. The Threat Landscape in 2021. – Symantec Threat Hunter Team. URL: https://www.software.broadcom.com/hubfs/SED/SED%20PDF%20Reports/The_Threat_Landscape_2021_12.pdf (дата звернення 05.11.2023).

6. Інформаційна безпека. URL: https://uk.wikipedia.org/wiki/Інформаційна_безпека (дата звернення 05.11.2023).

7. Gorbenko, A., Kharchenko, V., Tarasyuk, O., & Furmanov, A. (2006). F (I) MEA-technique of web services analysis and dependability ensuring. Rigorous Development of Complex Fault-Tolerant Systems, 153-167.

8. Стрілець, А. М. (2020). Аналіз процесного підходу до аудиту Linux-подібних серверних операційних систем.

9. Одарущенко, О. Б., & Запорожець, О. Ю. (2015). Програмна розробка імітаційнного моделювання гарантоздатних web-сервісів з урахуванням дефектів і вразливостей компонент.

10. Сальник, С. В., Сторчак, А. С., & Крамський, А. Є. (2019). Аналіз вразливостей та атак на державні інформаційні ресурси, що обробляються в

інформаційно-телекомунікаційних системах. Системи обробки інформації, (2 (157)), 121-128.

11. Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., & Roytman, M. (2021). Exploit prediction scoring system (epss). Digital Threats: Research and Practice, 2(3), 1-17.

12. SUSE Linux Enterprise. URL: https://en.wikipedia.org/wiki/SUSE_Linux_Enterprise (дата звернення 20.11.2023).

13. Ядро Linux. URL: https://uk.wikipedia.org/wiki/Ядро_Linux (дата звернення 20.11.2023).

14. macOS. UR: https://uk.wikipedia.org/wiki/MacOS (дата звернення 05.11.2023).

15. Red Hat Enterprise Linux. URL: https://en.wikipedia.org/wiki/Red_Hat_Enterprise_Linux MacOS (дата звернення 05.11.2023).

16. Windows Server 2008. URL: https://en.wikipedia.org/wiki/Windows_Server_2008 (дата звернення 05.11.2023).

17. Garcia, M., Bessani, A., Gashi, I., Neves, N., & Obelheiro, R. (2011, June). OS diversity for intrusion tolerance: Myth or reality?. In 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN) (pp. 383-394). IEEE.

18. Gorbenko, A., Romanovsky, A., Tarasyuk, O., & Biloborodov, O. (2019). From analyzing operating system vulnerabilities to designing multiversion intrusion-tolerant architectures. IEEE Transactions on Reliability, 69(1), 22-39.

19. NVD Data Feeds. URL: https://nvd.nist.gov/vuln/data-feeds (дата звернення 20.11.2023).

20. Exploits + Shellcode + GHDB. URL: https://gitlab.com/exploit-database/exploitdb (дата звернення 20.11.2023).

21. ETL. URL: https://uk.wikipedia.org/wiki/ETL (дата звернення 20.11.2023).

22. Сховище даних. URL: https://uk.wikipedia.org/wiki/Сховище_даних (дата звернення 20.11.2023).

23. Stars: A Pattern Language for Query Optimized Schemaю URL: https://c2.com/ppr/stars.html (дата звернення 20.11.2023).

24. Схема сніжинки. URL: https://uk.wikipedia.org/wiki/Схема_сніжинки (дата звернення 21.11.2023).

25. A Complete Guide to the Common Vulnerability Scoring System. URL: https://www.first.org/cvss/v2/guide (дата звернення 20.11.2023).

26. Common Vulnerability Scoring System v3.1: Specification Document. URL: https://www.first.org/cvss/v3.1/specification-document (дата звернення 21.11.2023).

27. Official Common Platform Enumeration (CPE) Dictionary. URL: https://nvd.nist.gov/products/cpe (дата звернення 21.11.2023).

28. Guide to Enterprise Patch Management Planning. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r4.pdf (дата звернення 05.11.2023).

29. IT Security Procedural Guide: Vulnerability Management Process CIO-IT Security-17-80. URL: https://www.gsa.gov/system/files/Vulnerability - Management-Process-%5BCIO-IT-Security-17-80-Rev-4%5D-03-13-2023.pdf (дата звернення 21.11.2023).

30. BOD 19-02: Vulnerability Remediation Requirements for Internet-Accessible Systems. URL: https://www.cisa.gov/news-events/directives/bod-19-02-vulnerability-remediation-requirements-internet-accessible-systems (дата звернення 21.11.2023).

31. Announcing .NET 8. URL: https://devblogs.microsoft.com/dotnet/ announcing-dotnet-8/ (дата звернення 21.11.2023).

32. ETLBox - a lightweight ETL library for .NET/C#. URL: https://www.etlbox.net/ (дата звернення 21.11.2023).

33. GitHub - Cinchoo/ChoETL: ETL framework for .NET (Parser / Writer for CSV, Flat, Xml, JSON, Key-Value, Parquet, Yaml, Avro formatted files). URL: https://github.com/Cinchoo/ChoETL (дата звернення 21.11.2023).

34. ETL.Net. Mass processing data finally easy for .NET developers. URL: https://paillave.github.io/Etl.Net/ (дата звернення 21.11.2023).

35. How to read JSON as .NET objects (deserialize). URL: https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/deserialization?pivots=dotnet-8-0 (дата звернення 21.11.2023).

36. Etl.Net/LICENSE at master · paillave/Etl.Net · GitHubю URL: https://github.com/paillave/Etl.Net/blob/master/LICENSE (дата звернення 21.11.2023).

37. Looker Studio Overview. URL: https://lookerstudio.google.com/overview (дата звернення 21.11.2023).

38. Metabase | Business Intelligence, Dashboards, and Data Visualization. URL: https://www.metabase.com/ (дата звернення 21.11.2023).

39. Welcome | Superset. URL: https://superset.apache.org/ (дата звернення 21.11.2023).

40. Choosing an Analytics Tool. Metabase Vs Superset Vs Redash. URL: https://medium.com/vortechsa/choosing-an-analytics-tool-metabase-vs-superset-vs-redash-afd88e028ba9 (дата звернення 21.11.2023).

41. The stream of events, a global picture. URL: https://paillave.github.io/Etl.Net/docs/quickstart/principle (дата звернення 20.11.2023).